Développement d'une plateforme de caractérisation rapide pour les qubits de spin

par

Marc-Antoine Roux

Mémoire présenté au département de physique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ des SCIENCES
UNIVERSITÉ de SHERBROOKE

Sherbrooke, Québec, Canada, 4 août 2020

Le 4 août

le jury a accepté le mémoire de Monsieur Marc-Antoine Roux dans sa version finale.

Membres du jury

Professeur Michel Pioro-Ladrière
Directeur de recherche
Département de physique

Professeur Claude Bourbonnais
Membre interne
Département de physique

Professeur Denis Morris
Président rapporteur
Département de physique

À ma famille et mes amis

Sommaire

L'initialisation de qubits de spin est complexe en raison de la variabilité dans la fabrication des boîtes quantiques et du grand nombre de grilles électrostatiques à ajuster avec précision. Ainsi, la mesure de nombreux diagrammes de stabilité est nécessaire pour connaître les tensions à utiliser pour créer une boîte quantique et aussi connaître le nombre d'électrons présents à l'intérieur de cette dernière. Puisque la mesure d'un diagramme de stabilité est généralement longue, il s'agit d'un facteur limitant dans l'initialisation de qubits de spin. Ainsi, une plateforme de caractérisation pour qubits de spin est présentée ici dans le but d'accélérer l'initialisation des qubits de spin. Cette plateforme est totalement modulaire pour permettre le développement de futurs processeurs quantiques avec un grand nombre de qubits et composée d'instruments de haute performance. À l'aide de celle-ci, la mesure d'un diagramme de stabilité a pu être faite jusqu'à 1000 fois plus rapidement comparée à la même mesure faite avec un instrument de laboratoire standard. Ce gain en vitesse permet également de faire des mesures en temps réel et, avec l'utilisation de grilles virtuelles, constitue une étape importante vers l'initialisation automatique des qubits de spin.

Remerciements

J'aimerais premièrement remercier mon superviseur Michel Pioro-Ladrière de m'avoir donné la chance de faire partie de son groupe. Évidemment, sans lui, je ne serais pas où j'en suis aujourd'hui. Michel est, selon moi, un vrai leader. Il a toujours les bons mots et l'enthousiasme pour nous motiver même quand notre projet ne va pas comme on le veut. Il a une vision et sait comment tirer le meilleur de chacun pour atteindre cette vision. J'aimerais également remercier Julien Camirand Lemyre et Sophie Rochette qui m'ont tous les deux supervisé durant ma maîtrise. Ils m'ont introduit aux boîtes quantiques et au travail en laboratoire. Leur rigueur et persévérance sont une source d'inspiration pour tous. Merci également à Alexandre Bédard-Vallée pour m'avoir assisté avec les transferts d'hélium, les soudures et la fabrication de PCB, ainsi qu'à Sara Turcotte qui m'a aidé à m'orienter dans le laboratoire à mon arrivée dans le groupe et qui m'a introduit à l'escalade. Je devrais aussi mentionner l'apport de Gregory Brooks qui m'a aidé à interpréter mes résultats quand mon circuit électrique est devenu plus complexe qu'à l'habitude. Merci aussi à Marc-Antoine Genest qui m'a aidé à plusieurs reprises avec la programmation HVI. Il était une des rares personnes qui savait en détail sur quoi je travaillais et ses commentaires étaient toujours précieux. Merci à Christian Lupien pour son aide avec la programmation FPGA et avec la résolution de mes problèmes de montage en général. Il s'est presque établi une routine où Christian passait en fin de journée pour voir les progrès de la veille. Son expertise et sa vue d'ensemble de mon projet m'ont sauvé énormément de temps quand je n'arrivais pas à trouver la source d'un des nombreux problèmes que j'ai rencontré. Ce projet a été fait en collaboration avec Keysight Technologies, alors je remercie les gens de Keysight pour leur soutien, en particulier Nizar Messaoudi, Julio Crespo et Gidget Heintz. Toute l'équipe était toujours disponible lorsque j'avais des questions sur un de leurs systèmes.

J'aimerais aussi remercier tous les autres membres du groupe pour les discussions intéressantes et les membres de l'Institut quantique qui font de l'IQ ce qu'il est aujourd'hui. Finalement, je voudrais reconnaître le soutien financier de l'Institut quantique tout au long

de ce projet. De plus, cette recherche a été menée grâce, en partie, au soutien financier du Fonds d'excellence en recherche Apogée Canada.

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

BQ Boîte quantique

MOS Métal-oxyde-semiconducteur

RF Radio-fréquence

EDSR Electric dipole spin resonance («résonance dipolaire électrique de spin» en français)

CNOT Controlled NOT gate (peut être traduit par «porte NON contrôlée» en français)

FPGA Field-programmable gate array

DC Direct current («courant continu» en français)

AC Alternating current («courant alternatif» en français)

QET Quantum engineering toolkit

PCIe Peripheral component interconnect express

PXI PCI eXtensions for instrumentation

AWG Arbitrary waveform generator («générateur de signaux arbitraires» en français)

HVI Hard virtual instrument

SMU Source measure unit

IPS Image par seconde

Chapitre 1

Introduction

Lorsque l'idée d'un ordinateur quantique a émergé à la fin du 20^{e} siècle, il n'était pas certain qu'une telle machine puisse un jour offrir plus de puissance de calcul qu'un ordinateur classique. La donne a changé lorsque Peter Shor a publié son algorithme de factorisation de nombres premiers [1]. Cet algorithme permet de factoriser un nombre en ses nombres premiers en temps polynomial, ce qui est presque qu'exponentiellement plus rapide que le meilleur algorithme de factorisation connu à ce jour [2]. Cette démonstration frappante a depuis motivé le développement d'un ordinateur quantique. Plusieurs grandes compagnies telles que Intel, Google, Microsoft et IBM se sont mises à investir massivement dans la recherche sur l'information quantique. Ces investissements ont mené, entre autres, au dévoilement du processeur quantique de Google, nommé Bristlecone, en 2018 au March meeting de l'American Physical Society [3]. Il s'agit d'une matrice carrée de 72 qubits supraconducteurs, un des plus gros processeurs quantiques à ce jour en ce qui concerne le nombre de qubits. Plus récemment, Google a affirmé avoir atteint la suprématie quantique, cette idée qu'un ordinateur quantique puisse faire un calcul qu'il est impossible de faire avec un ordinateur classique en un temps raisonnable, avec son plus récent processeur, Sycamore, possédant 54 qubits [4]. Bien que le nombre de qubits des processeurs quantiques développés au cours des dernières années ait augmenté rapidement, il reste encore beaucoup de chemin à faire pour espérer un jour pouvoir faire des calculs avec correction d'erreur. Il est estimé que pour avoir un qubit logique résistant aux erreurs, il faut autour de 1000 qubits physiques. Alors, pour exécuter un algorithme complexe, comme l'algorithme de Shor, et résoudre un problème intéressant, il faudrait de l'ordre d'un millier de qubits logiques soit un million de qubits physiques [5]. Pour cette raison, on peut considérer que le développement d'un ordinateur quantique n'en est encore qu'à ses débuts.

Ainsi, plusieurs architectures sont toujours en cours de développement puisqu'il n'est pas clair encore laquelle sera la plus adaptée aux besoins d'un ordinateur quantique avec des milliers de qubits, voire plus. Les qubits supraconducteurs [6] sont actuellement l'architecture la plus avancée grâce à leur facilité de fabrication. Il existe cependant d'autres architectures basées sur les ions piégés [7], les fermions de Majorana [8] et les qubits de spin [9][10]. L'avance des qubits supraconducteurs a motivé le développement d'électronique de contrôle adaptée aux besoins spécifiques de cette architecture. L'électronique de contrôle joue un rôle crucial dans la mise à l'échelle de chaque architecture et donc dans l'augmentation du nombre de qubits d'un processeur quantique. Par contre, l'utilisation d'électronique de contrôle spécialisée à une architecture particulière peut aussi grandement accélérer son développement. Par exemple, dans le cas des qubits de spin, une caractérisation approfondie est nécessaire puisque le rendement de fabrication est généralement faible et les propriétés des qubits varient d'un échantillon à l'autre à cause de défauts au niveau atomique. À chaque nouvelle expérience, il faut donc mesurer les caractéristiques d'intérêt avant de pouvoir utiliser les qubits à des fins de calculs quantiques. Étant donné que cette procédure est coûteuse en temps et répétée souvent, des instruments de mesure spécialisés pourraient grandement accélérer cette étape de développement. Ce projet de maîtrise porte donc sur ce dernier aspect.

Le contenu de ce mémoire est divisé en plusieurs parties de sorte à bien comprendre et identifier les opérations typiques faites avec des qubits de spin pour ensuite en déduire les besoins en électronique de contrôle. Ainsi, le chapitre 2 contient les définitions des termes techniques utilisés tout au long du mémoire, ainsi que les détails sur les expériences typiques avec les qubits de spin. À partir de ces expériences, les caractéristiques techniques d'une plateforme de caractérisation rapide pour qubits de spin sont énoncées au chapitre 3. Les chapitres 4, 5 et 6 présentent la plateforme choisie pour développer les routines de caractérisations pour qubits de spin, ainsi que les progrès qui ont été faits dans la programmation de ces différentes routines. Finalement, le chapitre 7 décrit les perspectives du projet.

Chapitre 2

Concepts

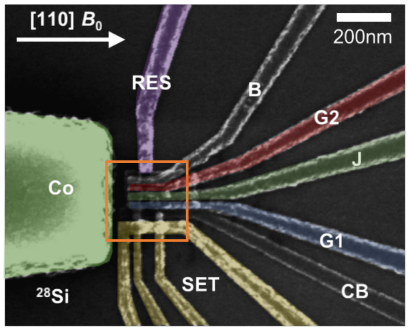
Pour bien comprendre les caractéristiques techniques nécessaires pour les instruments qui servent à faire des mesures avec des boîtes quantiques, il faut d'abord commencer par expliquer ce que sont les boîtes quantiques et comment les former. Il existe de nombreuses architectures de boîtes quantiques, mais les concepts vus ici sont généraux et s'appliquent à toutes.

2.1 Description d'une boîte quantique

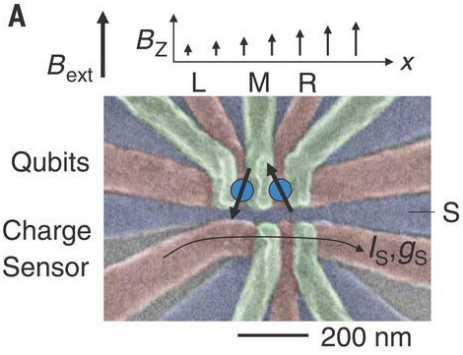
Le principe d'une boîte quantique est d'emprisonner un ou plusieurs porteurs de charge (électrons ou trous) dans un puits de potentiel tridimensionnel. Dans le cas présent, ce sont des électrons qui sont utilisés. Les boîtes quantiques sont conçues habituellement avec plusieurs grilles métalliques qui appliquent des potentiels électrostatiques tout autour d'un point dans l'espace de sorte à former un puits de potentiel. Une fois le confinement électrostatique présent, il est possible de l'abaisser de façon à laisser un électron entrer dans la boîte quantique ou, inversement, de l'augmenter pour éjecter un électron de la boîte quantique. Le mouvement d'électrons à travers une boîte quantique se fait par effet tunnel, ainsi le potentiel électrostatique autour de cette dernière a un grand effet sur la probabilité qu'ont les électrons d'entrer ou sortir. Il s'agit ici d'une description très schématisée d'une boîte quantique. Pour bien saisir leur complexité, il est nécessaire d'aller dans plus de détails. Premièrement, il est nécessaire d'avoir plusieurs électrons libres autour de la boîte quantique pour pouvoir en piéger. Cela est typiquement fait en choisissant les bonnes couches de matériaux pour le substrat des échantillons. Par exemple, une interface de GaAs/AlGaAs ou de Si/SiGe crée un potentiel attractif pour les électrons à l'interface des

deux couches. Les électrons sont alors confinés en 2D à cette interface, ce qui crée un gaz d'électrons bidimensionnel. Le même résultat peut aussi être atteint avec un semiconducteur dopé comme dans les dispositifs métal-oxyde-semiconducteur (MOS). Une fois que le confinement en 2D est créé par le substrat, il est nécessaire de déposer des grilles métalliques à la surface du substrat pour former la boîte quantique en tant que telle. Les grilles permettent d'appliquer des potentiels électrostatiques au niveau du gaz d'électrons pour le déformer. Pour cette raison, ces grilles sont aussi appelées grilles électrostatiques. Avec une bonne géométrie de grilles, il est possible de contrôler précisément le nombre d'électrons dans un point de l'espace. On obtient ainsi une boîte quantique. Pour pouvoir facilement remplir ou vider une boîte quantique, d'autres grilles sont utilisées pour créer ce qu'on appelle des réservoirs d'électrons de part et d'autre de la boîte quantique. Généralement un réservoir, que l'on appelle la source, sert à injecter des électrons dans la boîte quantique, alors qu'un autre réservoir, le drain, sert à vider la boîte quantique. En pratique, on peut considérer ces réservoirs comme d'énormes boîtes quantiques contenant beaucoup d'électrons. S'il y a des niveaux d'énergie inoccupés dans la boîte quantique, il sera possible pour un électron d'un réservoir de remplir ce niveau. Pour contrôler l'occupation électronique, il faut contrôler précisément le potentiel chimique à l'intérieur de la boîte. C'est la barrière de potentiel entre les réservoirs et la boîte quantique qui permet de contrôler avec précision le flux d'électrons entrant ou sortant de la boîte quantique. La forme de la barrière de potentiel dicte le taux tunnel qui est la fréquence moyenne à laquelle un électron passe d'un réservoir à la boîte quantique ou vice-versa. L'inverse du taux tunnel donne le temps moyen après lequel un électron sort ou entre dans la boîte quantique. Il s'agit d'une métrique importante pour la mesure des états de spin dans une boîte quantique.

Le nombre de grilles à contrôler pour créer une boîte quantique varie selon l'architecture utilisée, mais est généralement autour de 4 grilles. La figure 2.1 montre plusieurs arrangements de grilles pouvant former des boîtes quantiques. Avec 4 grilles à ajuster par boîte quantique, il est facile de comprendre que le nombre de grilles augmente rapidement si plusieurs boîtes quantiques sont utilisées pour une expérience. Bien sûr, trouver les valeurs de tension à appliquer sur chaque grille pour obtenir le profil de potentiel optimal n'est pas une tâche facile. La technique pour trouver ces valeurs est de faire des diagrammes de stabilité.

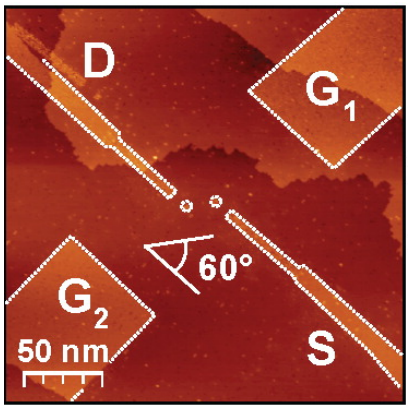


(**a**) Double boîte quantique Si-MOS [11]



(**b**) Double boîte quantique en SiGe [12]

(**c**) Boîte quantique avec une géométrie de grille divisée («split gate» en anglais) [13]



(**d**) Double boîte quantique avec donneurs [14]

FIGURE 2.1 Différentes architectures de boîtes quantiques

2.2 Diagrammes de stabilité

Un diagramme de stabilité est un graphique 2D qui décrit l'occupation électronique d'une ou plusieurs boîtes quantiques en fonction de la tension appliquée sur les grilles autour de ces boîtes quantiques. Typiquement, deux grilles sont utilisées comme axes du diagramme pour étudier l'effet de ces grilles sur le nombre d'électrons dans les boîtes quantiques. Quelques exemples de diagrammes de stabilité sont donnés à la figure 2.2. Plusieurs diagrammes peuvent être mesurés avec différents couples de grilles pour obtenir l'effet de toutes les grilles. Le signal d'un diagramme de stabilité peut venir de mesures en transport, où le courant qui traverse la boîte donne de l'information sur l'occupation, ou d'un détecteur de charge, où la capacité de la boîte quantique, qui agit comme un condensateur, indique combien d'électrons sont dans la boîte. Les lignes de transition de charge d'un diagramme viennent, comme leur nom l'indique, d'un déplacement de charge dans la boîte quantique. Expérimentalement, un déplacement de charge se voit à travers un changement de courant ou de capacité dans la boîte quantique. C'est pourquoi la dérivée du signal est souvent utilisée pour mettre en évidence les transitions. Il est à noter qu'un détecteur de charge a l'avantage d'être plus sensible que les mesures en transport dans le régime à un électron, mais est généralement plus complexe à utiliser. Une fois qu'un diagramme de stabilité est obtenu, il est possible d'ajuster les tensions de la boîte quantique pour y choisir la configuration électronique et en faire un qubit. Pour former un qubit de spin, il faut généralement avoir seulement un électron dans une ou plusieurs boîtes quantiques [16]. Il est toutefois possible qu'une architecture utilise plusieurs électrons dans une seule boîte quantique [11]. Le point important ici est de réaliser que pour former un qubit de spin, il faut pouvoir contrôler précisément le nombre d'électrons dans plusieurs boîtes quantiques.

2.3 Qubits de spin

Une fois que plusieurs boîtes quantiques ont été créées et qu'elles contiennent un nombre connu d'électrons, il est possible d'en faire des qubits de spin. L'arrangement des boîtes quantiques, leur nombre et le nombre d'électrons dans chaque boîte détermine le type de qubit qui peut être créé. Il est possible d'encoder de l'information dans le spin d'un seul électron [17], dans des états singulet et triplet [18] et même dans d'autres configurations comme le qubit d'échange [19]. Toutefois, seulement les qubits avec un seul électron par boîte quantique seront considérés dans ce mémoire pour simplifier la discussion. Les concepts qui seront expliqués avec ce type de qubit particulier s'appliquent cependant à tous les autres types. Les explications restent donc tout à fait générales. De plus, les caractéristiques

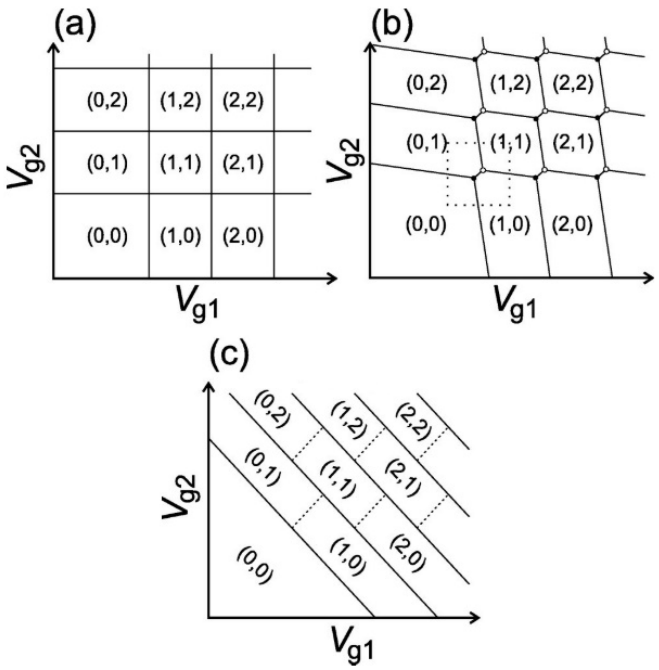
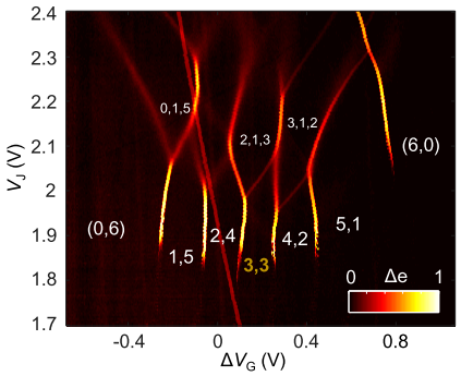


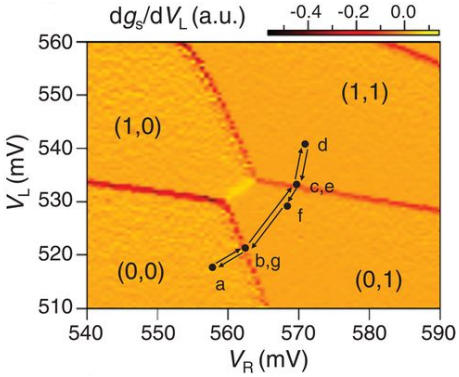
FIGURE 2.2 Différents types de diagramme de stabilité selon le couplage entre deux boîtes quantiques. Dans cet exemple, les tensions des grilles 1 et 2 (V_{g1} et V_{g2}) sont variées pour donner les diagrammes de stabilité. La notation (1,0) indique qu'il y a un électron dans la première boîte quantique et aucun dans la deuxième. En a), il n'y a pas de couplage entre les deux boîtes quantiques. En b), il y a un couplage entre les boîtes quantiques. Un changement de tension sur la grille de la boîte 1 change à quelle tension de la grille 2 on peut observer une transition. En c), les deux boîtes quantiques sont fusionnées, ce qui représente le cas de couplage maximal. La figure a été adaptée de la référence [15].

techniques nécessaires pour opérer un qubit de spin sont aussi similaires d'un type à l'autre. Ces caractéristiques techniques sont présentées à la section 3.

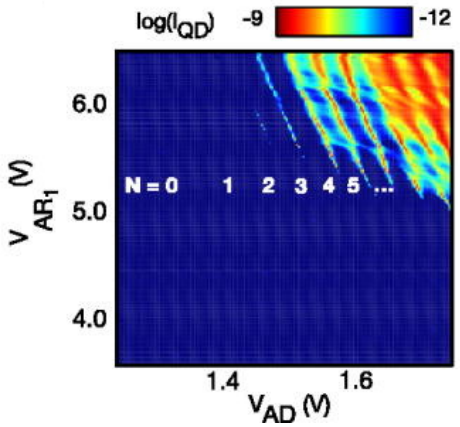
Pour former un qubit de spin, il faut premièrement appliquer un champ magnétique externe sur les boîtes quantiques pour créer une séparation Zeeman. Ainsi, les états de spin parallèle $|\uparrow\rangle$ et antiparallèle $|\downarrow\rangle$, par rapport à la direction du champ magnétique, n'ont plus la même énergie. Ce sont ces deux états physiques qui permettent d'encoder les états logiques 0 et 1. En plus de ce champ externe, un champ magnétique oscillant doit pouvoir être généré dans les boîtes quantiques pour permettre de faire des rotations de spin. Ce sont ces rotations de spin qui permettent de faire des opérations logiques.



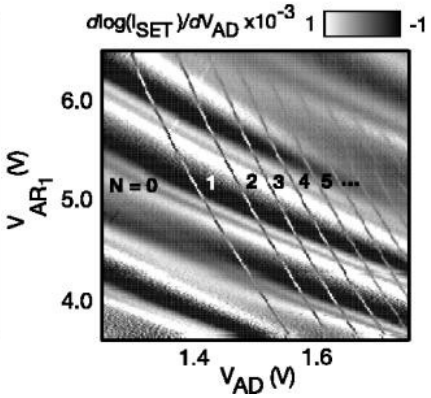
(**a**) Double boîte quantique Si-MOS [11]



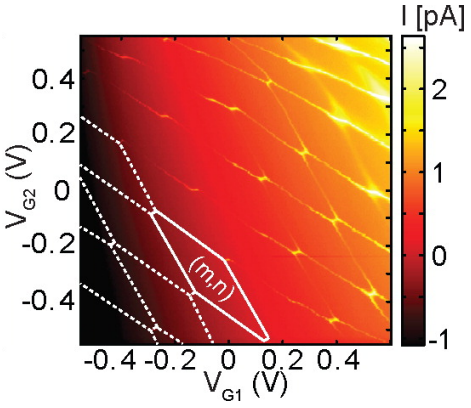
(**b**) Double boîte quantique en SiGe [12]



(**c**) Boîte quantique avec une géométrie de grille divisée (mesure en transport) [13]



(**d**) Boîte quantique avec une géométrie de grille divisée (mesure par détection de charge) [13]



(**e**) Double boîte quantique avec donneurs [14]

FIGURE 2.3 Différents diagrammes de stabilité pour les différents types de boîtes quantiques de la figure 2.1

2.4 Opérations logiques sur qubits de spin

Cette section sert à donner quelques exemples d'opérations logiques qu'il est possible d'appliquer sur un qubit, ainsi que les étapes expérimentales nécessaires.

2.4.1 Opérations à un qubit

L'opération la plus simple pour un spin est l'inversion de bit, c'est-à-dire passer de l'état de spin antiparallèle à l'état de spin parallèle ou vice-versa. Pour faire cette opération, on utilise la résonance de spin électronique qui consiste à utiliser un champ magnétique oscillant à une fréquence spécifique pour faire basculer le spin de l'électron. Plus précisément, un champ magnétique externe (B_{ext}) est utilisé pour créer une séparation Zeeman entre les états de spin parallèle et antiparallèle de l'électron. Il est assumé ici que l'état de spin antiparallèle est l'état fondamental du système et que l'état de spin parallèle est l'état excité. Cette séparation énergétique (E_z) est donnée par l'équation 2.1.

$$E_z = \hbar\omega_0 = g\mu_B B_{ext} \tag{2.1}$$

Dans la dernière équation, on définit $\hbar$ comme la constante de Planck réduite, g comme le facteur de Landé, et μ_B comme le magnéton de Bohr. Si un champ magnétique oscillant (B_{osc}) perpendiculaire au champ externe est appliqué à la fréquence ω_0, donnée par la séparation Zeeman, le système entre en résonance et le spin de l'électron se met à tourner à une fréquence donnée par l'équation 2.2. Il est à noter que le champ magnétique oscillant utilisé a typiquement une polarisation linéaire qui se décompose en deux polarisations circulaires où l'une est horaire et l'autre antihoraire. Étant donné que seulement une des deux polarisations circulaires est en résonance avec la précession de Larmor du spin, le champ oscillant est effectivement réduit de moitié. Dans les équations, le champ oscillant utilisé est donc $B_{osc}/2$.

$$\omega = \frac{g\mu_B B_{osc}}{2\hbar} \tag{2.2}$$

Ainsi, pour inverser le spin, il suffit d'appliquer le champ magnétique oscillant pendant un certain temps de sorte à effectuer une rotation de π radians sur la sphère de Bloch. L'angle de rotation θ est donné par $\theta = \omega t$. Alors, le temps nécessaire pour inverser le spin est donné par l'équation 2.3.

$$t = \frac{\theta}{\omega} = \frac{2\hbar\pi}{g\mu_B B_{osc}} = \frac{h}{g\mu_B B_{osc}} \tag{2.3}$$

On remarque que plus l'amplitude du champ oscillant est élevée, plus la rotation est rapide. Un grand champ magnétique oscillant est donc crucial pour effectuer des opérations logiques rapides. Une façon de générer un champ magnétique oscillant est d'utiliser l'induction électromagnétique et d'injecter un courant oscillant dans une ligne à transmission pour micro-ondes [20]. Cette technique a cependant le désavantage de dissiper de la chaleur, ce qui peut réduire le temps de cohérence des qubits. En plus, la ligne à transmission occupe beaucoup d'espace sur l'échantillon, ce qui ajoute des contraintes lors de la conception des dispositifs. Il est donc plus avantageux d'utiliser des microaimants pour créer un gradient de champ magnétique dans la boîte quantique et d'y déplacer l'électron avec des impulsions électriques pour générer un champ magnétique oscillant effectif [21]. Cette dernière méthode se nomme la résonance dipolaire électrique de spin ou «Electric dipole spin resonance» (EDSR) en anglais.

2.4.2 Opérations à deux qubits

Pour les opérations à deux qubits, la procédure est similaire aux opérations à un qubit, mais il faut y ajouter la manipulation de l'interaction entre qubits. En effet, pour les manipulations à un qubit, il est nécessaire d'isoler chaque qubit de sorte à ne pas en affecter plus d'un. Au contraire, pour les manipulations à deux qubits, il faut permettre aux deux qubits d'intérêt à interagir de façon contrôlée. Comme exemple, la porte logique CNOT [1] résonante sera considérée [12]. Pour faire une porte CNOT, il faut faire une rotation sur un qubit conditionnelle à l'état d'un 2e qubit. Plus précisément, si le qubit de contrôle est dans l'état $|1\rangle$, le qubit cible est inversé. Le circuit d'une porte CNOT, ainsi que les impulsions électriques à utiliser pour implémenter cette dernière dans un système à deux qubits de spin, sont donnés aux figures 2.4 a) et b) respectivement. Dans un système avec une double boîte quantique, c'est possible d'y arriver en jouant avec l'interaction d'échange (J) entre les deux boîtes. Lorsque l'interaction est nulle, les deux qubits peuvent être manipulés indépendamment. Comme on le voit à la figure 2.4 c), la fréquence f_L pour effectuer une rotation, représentée par les flèches entre deux niveaux d'énergie, sur le qubit de gauche (en rouge) est la même peu importe si l'état initial est $|\downarrow\downarrow\rangle$ ou $|\uparrow\uparrow\rangle$. Le même constat peu être fait avec le qubit de droite (en bleu), excepté que f_L et f_R sont légèrement différentes à cause du gradient appliqué sur les deux boîtes quantiques. Par contre, lorsque l'interaction d'échange est augmentée en diminuant la hauteur de la barrière tunnel entre les deux qubits, la fréquence utilisée pour passer de l'état $|\uparrow\uparrow\rangle$ à $|\downarrow\uparrow\rangle$ n'est plus la même que pour passer de l'état $|\downarrow\downarrow\rangle$ à $|\uparrow\downarrow\rangle$. Cela est possible à condition que le gradient de champ magnétique, δB,

1. CNOT est un acronyme en anglais pour «controlled NOT gate» qui peut être traduit par «porte NON contrôlée»

soit beaucoup plus grand que J. Lorsque cette condition est respectée, les états de spin $|\downarrow\uparrow\rangle$ et $|\uparrow\downarrow\rangle$ deviennent plus faibles en énergie d'une quantité $J/2$ par rapport aux états de spin $|\uparrow\uparrow\rangle$ et $|\downarrow\downarrow\rangle$. Ce phénomène peut être compris avec le principe d'exclusion de Pauli. Plus les électrons des deux boîtes quantiques interagissent, plus il y aura une répulsion si les électrons ont le même spin. Au contraire, les états de spin différents sont énergétiquement favorisés puisqu'il n'y a pas de répulsion. Alors, il est possible, par exemple, d'effectuer une rotation sur le spin de gauche qui est conditionnelle à l'état du spin de droite si le champ magnétique oscillant est envoyé à la fréquence $f^{L}_{|\Psi_R\rangle=|\uparrow\rangle}$.

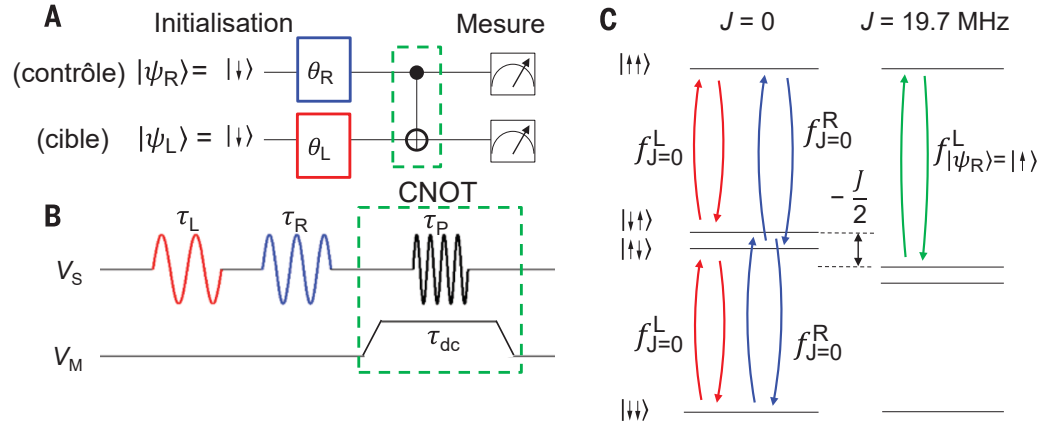


FIGURE 2.4 Fonctionnement d'un CNOT résonant. a) Circuit de la porte CNOT. La porte CNOT est encadrée en vert. Les qubits contrôle et cible sont initialisés dans l'état $|\downarrow\rangle$. Ensuite, une rotation d'un angle variable θ est faite sur chaque qubit pour tester la porte avec différents états. b) Tensions de grilles et impulsions nécessaires à l'implémentation expérimentale du circuit en a). Les temps τ_L, τ_R correspondent aux temps nécessaires pour effectuer les rotations à un qubit d'un angle θ_L et θ_R respectivement. Ces rotations sont générées avec la méthode EDSR en variant la tension de grille V_s. La figure 2.1b présente une image du dispositif où les grilles d'intérêt sont identifiées. La tension de grille V_M sert, quant à elle, à modifier le couplage entre les deux boîtes quantiques pour permettre la rotation conditionnelle sur le qubit cible. Les temps d'impulsions τ_p et τ_{dc} servent à calibrer la porte CNOT. c) Niveaux d'énergie de la double boîte quantique selon le couplage J entre les deux boîtes. Le niveau d'énergie fondamental a été modifié de $|\uparrow\uparrow\rangle$ à $|\downarrow\downarrow\rangle$ par rapport à la version originale de la figure pour corriger ce qui semble être une erreur. De plus, la figure a été tirée de la référence [12], puis traduite en français.

2.4.3 Lecture du spin

Une fois que les opérations voulues ont été faites, il faut lire l'état de spin final. Cependant, le moment magnétique d'un seul électron est très petit et donc difficile à mesurer. Une astuce à utiliser est plutôt de transférer l'information de spin dans un état de charge et de mesurer

la charge électrique à la place. Cette méthode s'appelle la conversion spin à charge. Dans des boîtes quantiques, deux méthodes sont principalement utilisées soit la mesure sélective en énergie et la mesure sélective en taux tunnel [22]. Ces deux méthodes sont illustrées à la figure 2.5. La mesure sélective en énergie utilise le fait que certaines transitions ne sont pas énergétiquement permises. Par exemple, en présence d'un champ magnétique, les états de spin parallèle et antiparallèle ne sont pas à la même énergie. Ainsi, si le potentiel chimique d'un réservoir servant à vider la boîte quantique peut être positionné entre l'état fondamental et excité de spin. Dans cette configuration, l'électron ne peut sortir de la boîte quantique que si ce dernier est dans l'état excité de spin. L'état de spin peut alors être déduit en mesurant le courant sortant de la boîte quantique à l'aide d'un détecteur de charge.

La mesure sélective en taux tunnel, quant à elle, tire profit du taux tunnel qui est différent pour l'état excité (Γ_e) et fondamental (Γ_f) afin de différencier les deux états. Dans ce cas-ci, le potentiel chimique du réservoir est placé sous le niveau d'énergie des deux états de spin. La mesure du courant est faite jusqu'à un temps τ qui est plus grand que le temps moyen pour un électron de quitter l'état excité $\tau_e = 1/\Gamma_e$, mais plus petit que le temps moyen pour un électron de quitter l'état fondamental $\tau_f = 1/\Gamma_f$. Ainsi, si une augmentation de courant est mesurée entre le début de la mesure et le temps τ écoulé, on en déduit que l'électron était dans l'état excité. Par contre, pour que cette méthode soit efficace, il faut respecter la condition $\tau_f \ll \tau \ll \tau_e$. Lorsque l'architecture le permet, il est même possible de combiner les deux méthodes de mesure pour augmenter la fidélité et la rapidité de la mesure [23]. Il faut cependant avoir un couplage tunnel entre la boîte quantique et le détecteur de charge utilisé.

2.5 État de l'art pour l'atteinte du régime à un électron dans les boîtes quantiques

L'atteinte du régime à un électron dans des boîtes quantiques est souvent faite manuellement étant donné la complexité de la tâche, mais il y a de plus en plus d'algorithmes créés pour effectuer automatiquement cette tâche. Plusieurs projets à l'Institut quantique se sont penchés sur ce problème. Par exemple, Maxime Lapointe-Major a développé un algorithme de reconnaissance d'image basée sur la transformée de Hough pour identifier les transitions d'une boîte quantique et déterminer les tensions à ajuster pour atteindre le régime à un électron (figure 2.6 a). Andrew Mounce, chez Sandia National Labs, a aussi travaillé sur une méthode similaire en utilisant la transformée de Hough généralisée pour identifier les transitions d'une double boîte quantique [24]. De plus, Marc-Antoine Genest a aussi

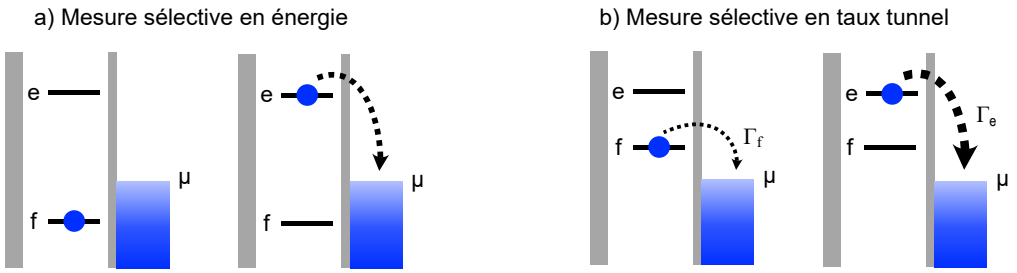
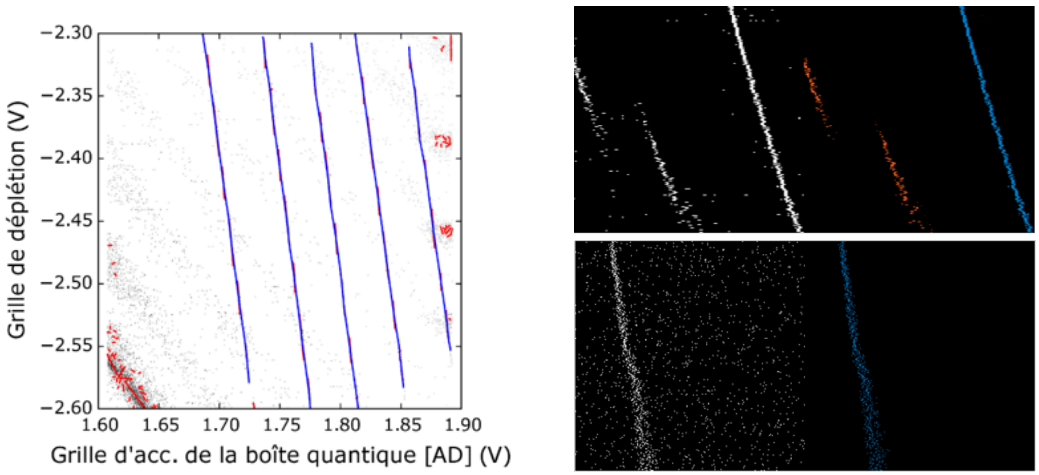


FIGURE 2.5 Comparaison de la mesure sélective en énergie en a) et en taux tunnel en b). La mesure sélective en énergie tire profit du fait que la transition de la boîte quantique vers le réservoir est interdite pour un électron dans l'état fondamental puisque le potentiel chimique du réservoir est plus haut que le potentiel chimique de la boîte. Si un électron sortant de la boîte quantique est détecté, on en déduit donc que ce dernier était dans l'état excité. La mesure sélective en taux tunnel utilise plutôt le fait que l'état excité de la boîte quantique a un taux tunnel vers le réservoir beaucoup plus grand que l'état fondamental. Ainsi, si un électron sortant de la boîte quantique est détecté entre le début de la mesure et un temps τ dépendant du taux tunnel de l'état fondamental, il est possible d'inférer que l'état de l'électron était dans l'état excité puisqu'un électron dans l'état fondamental n'aurait pas eu le temps de quitter la boîte quantique.

créé un algorithme permettant d'identifier les lignes de transition d'une boîte quantique en utilisant des méthodes d'apprentissage profond (figure 2.6 b). Il y a également plusieurs groupes faisant partie de la communauté des qubits de spin qui ont travaillé sur l'automatisation des méthodes permettant d'atteindre le régime à un électron dans les boîtes quantiques. Notamment, cette tâche a été accomplie dans une double boîte quantique avec un algorithme nécessitant seulement l'arrangement des grilles du dispositif utilisé et la valeur de tension de pincement de la grille T dans la figure 2.7. La tension de pincement fait référence ici à la tension nécessaire pour couper le courant circulant dans la rangée de boîtes quantiques. Ainsi, on «pince» le canal de conduction jusqu'à ce qu'il soit complètement fermé. D'autres travaux ont aussi démontré qu'il est possible de mesurer en priorité les parties d'un diagramme où il y a de l'information pertinente, ce qui permet de réduire le nombre de mesures à effectuer et donc le temps d'acquisition. L'algorithme, décrit à la figure 2.8, fait un balayage à faible résolution et, grâce à un modèle génératif profond, permet de reconstruire le diagramme de haute résolution pour identifier les parties importantes à mesurer. Dans ce cas-ci, des diamants de Coulomb ont été mesurés et la partie du diagramme qui est inutile est la zone où il n'y a pas de courant. Puisqu'il peut être long de balayer une grande plage de tension avec une haute résolution et de mesurer plusieurs fois la même zone pour faire du moyennage, la possibilité de réduire le nombre de mesure à faire peut se traduire par une réduction significative du temps de mesure. Les exemples précédents permettent d'accélérer la caractérisation des dispositifs qui est nécessaire pour atteindre le régime à un électron en exploitant le fait qu'une machine est habituellement plus rapide



(**a**) Algorithme utilisant la transformée de Hough [25]

(**b**) Algorithme utilisant l'apprentissage profond [26]

FIGURE 2.6 Identification des transitions d'une boîte quantique avec différentes méthodes. a) L'algorithme identifie le signal mesuré en rouge et relie en bleu les segments de lignes de transition de charge. b) L'algorithme peut non seulement identifier les lignes de transition de charge, mais il peut aussi distinguer différentes lignes qui n'appartiennent pas à la même transition. Les différentes lignes ont donc une couleur différente. Pour les deux exemples du haut et du bas, les données brutes sont à gauche et les données analysées sont à droite.

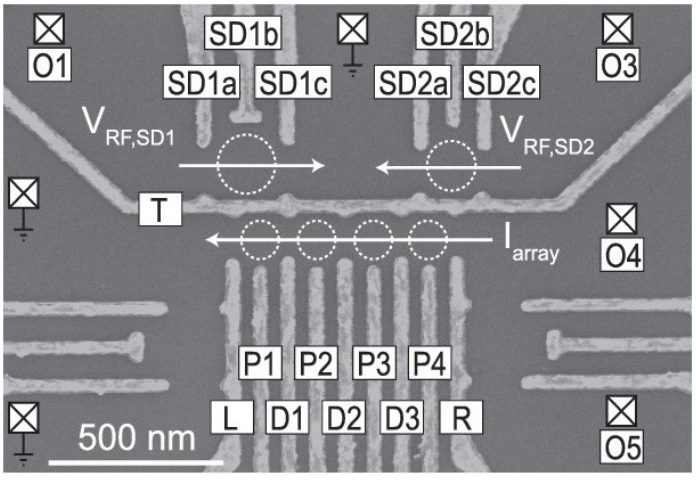


FIGURE 2.7 Dispositif utilisé pour former une double boîte quantique automatiquement [27]. Les grilles P1 à P4 servent à contrôler le nombre d'électrons dans chaque boîte quantique, alors que les grilles D1 à D3 servent à contrôler la barrière tunnel entre chaque boîte quantique. Les grilles L et R, quant à elles, contrôlent les barrières tunnel entre les boîtes quantiques et les réservoirs de gauche et de droite respectivement. Finalement, la grille T aide à la formation des boîtes quantiques et les grilles au-dessus servent à former des détecteurs de charge pour faire la mesure de diagrammes de stabilité.

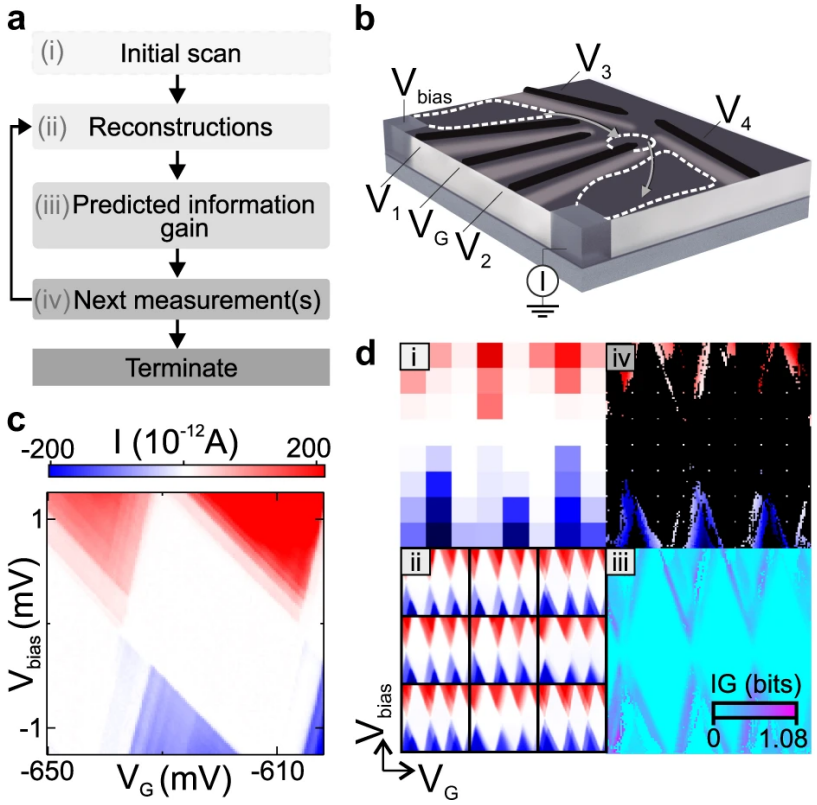


FIGURE 2.8 Mesures partielles d'un diagramme grâce à un algorithme basé sur un modèle génératif profond [28]. a) Description de l'algorithme. Une mesure rapide à basse résolution est effectuée (i) et envoyée au modèle génératif profond pour tenter de reconstruire (ii) la mesure complète à faire. À partir des reconstructions, les prochaines mesures à faire sont déterminées en fonction de leur gain estimé en information (iii). Une fois ces nouvelles mesures complétées (iv), l'algorithme retourne à l'étape (i) jusqu'à ce que le critère de fin soit atteint. b) Image du dispositif utilisé pour tester l'algorithme. c) Exemple de mesure complète avec une haute résolution. d) Représentation visuelle de l'algorithme décrit en a).

qu'une personne pour effectuer certaines tâches si un algorithme performant est utilisé. Par contre, une autre façon d'accélérer la caractérisation est de tout simplement utiliser des instruments plus performants. En poursuivant cette stratégie, le groupe de Jason Petta a pu mesurer un diagramme de stabilité d'une double boîte quantique en temps réel grâce à un réseau de portes logiques programmables par effet de champ, désigné par l'acronyme anglais FPGA (field-programmable gate array) et un amplificateur paramétrique Josephson [29]. Avec un temps de mesure de 4 ms pour un diagramme de 100 x 100 points, il est possible de répéter en boucle les mesures pour obtenir un vidéo de ce qui se passe au niveau de l'échantillon en temps réel. Ainsi, dans cet article, le groupe a pu modifier manuellement une double boîte quantique pour en faire une boîte quantique simple et inversement en quelques minutes à peine. Vu le gain en temps de mesure qu'il est possible d'atteindre avec un FPGA, plusieurs groupes ont commencé à développer des systèmes de contrôle de qubits utilisant des FPGAs [30][31].

2.6 Séquence complète pour faire un calcul quantique

Maintenant que chaque étape importante servant à faire un calcul quantique avec des qubits de spin a été introduite, il est possible de définir un protocole pour faire un calcul avec deux qubits en commençant par l'initialisation de deux boîtes quantiques et en terminant par la lecture des spins. Le diagramme de stabilité sera utilisé pour illustrer chaque étape du calcul. Les différentes étapes sont détaillées dans les prochaines sous-sections et se rattachent à la figure 2.9.

2.6.1 Initialisation

Premièrement, il faut créer deux boîtes quantiques. Tel qu'il a été mentionné à la section 2.5, plusieurs algorithmes permettent maintenant d'effectuer cette tâche. Une fois créées, il faut ajouter un électron dans chaque boîte. Les deux boîtes sont nommées BQ1 et BQ2 et la notation (Nb d'électrons dans BQ1, Nb d'électrons dans BQ2) sera utilisée. Il est possible d'aller à la région (2,0) du diagramme de stabilité et d'attendre plus longtemps que le temps de relaxation pour être certain que les deux spins soient dans leur état fondamental, c'est-à-dire l'état singulet. Ensuite, en allant dans la région (1,1) de façon adiabatique, le théorème adiabatique dicte que le système restera dans son état fondamental, c'est-à-dire $|\downarrow\rangle_{BQ1}|\downarrow\rangle_{BQ2} = |\downarrow\downarrow\rangle$. L'initialisation est faite lors des étapes 1 à 5 de la figure 2.9. Il est aussi possible d'utiliser la même technique que la mesure sélective en énergie [12] pour injecter uniquement des spins antiparallèles dans les boîtes quantiques. En positionnant le potentiel

chimique d'un réservoir au centre des niveaux fondamental et excité, on est assuré que seulement les électrons avec un spin dans l'état fondamental peuvent entrer dans la boîte quantique.

2.6.2 Manipulation du spin

Pour manipuler le spin dans une seule boîte quantique, il faut envoyer une impulsion à la fréquence donnée par la séparation Zeeman dans cette boîte quantique. La grille utilisée pour envoyer l'impulsion dépend de l'arrangement de grilles propre au dispositif utilisé, mais la grille doit pouvoir faire déplacer l'électron dans le gradient de champ magnétique. Dans la figure 2.9, les spins des boîtes quantiques 1 et 2 sont inversés aux étapes 5.1 et 5.2 respectivement. Étant donné que les qubits n'ont pas la même séparation Zeeman à cause du gradient de champ magnétique, les impulsions orange et verte ne sont pas à la même fréquence. Elles affectent donc qu'un seul qubit.

Pour faire une porte à deux qubits (le CNOT sera utilisé comme exemple ici), il faudra combiner la séquence faite pour manipuler le spin d'un seul électron avec la séquence utilisée pour modifier l'interaction entre les deux qubits. Pour modifier l'interaction, il suffit d'augmenter ou baisser la tension appliquée sur la grille qui contrôle la barrière tunnel entre les qubits. Dans la figure 2.9, le CNOT est fait à l'étape 5.3.

2.6.3 Lecture du spin

Pour faire la lecture des spins, il faut traverser minutieusement les transitions (1,1)-(0,1) (étape 6 de la figure 2.9) et (0,1)-(0,0) (étape 7 de la la figure 2.9) du diagramme de stabilité en contrôlant avec précision les potentiels chimiques des boîtes quantiques et des réservoirs. Ainsi, on vide les boîtes quantiques une à la fois pour y mesurer l'état de spin. La détection (ou non) d'un électron quittant la boîte quantique d'intérêt pendant la séquence de mesure permet d'inférer l'état de spin.

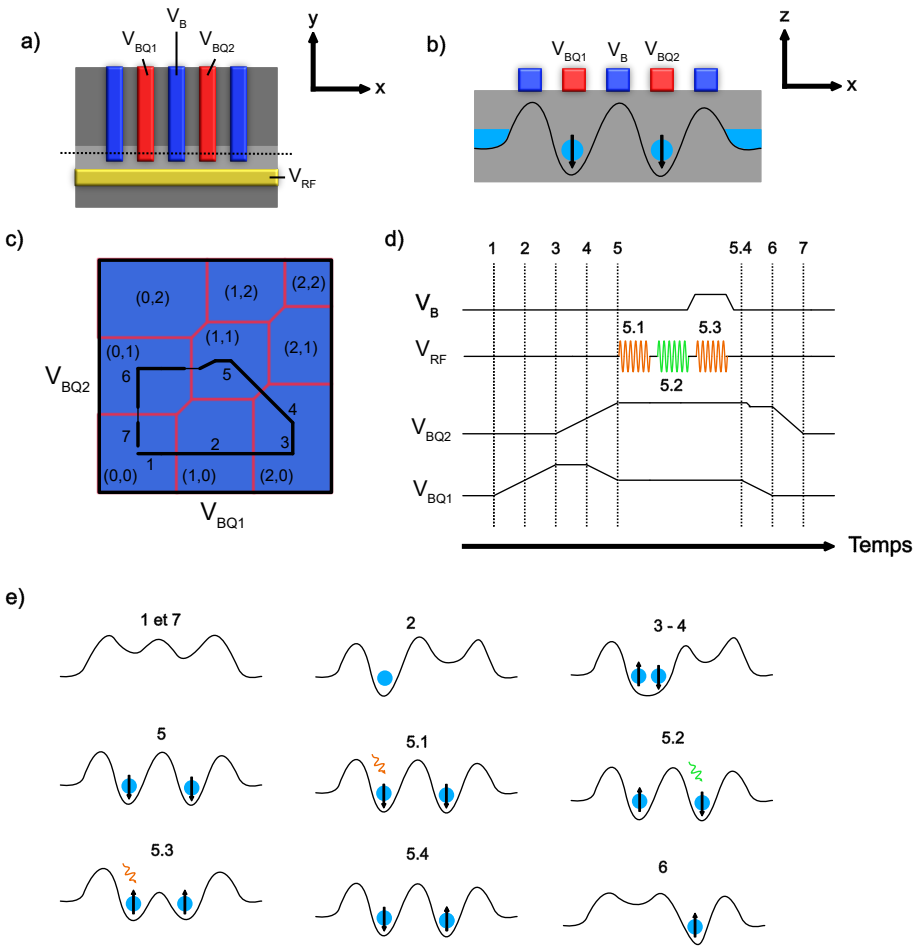


FIGURE 2.9 Séquence de calcul quantique. a) Vue du haut d'un dispositif fictif utilisé pour simuler une double boîte quantique. Il est important de noter qu'un champ magnétique externe en z est appliqué pour créer la séparation Zeeman. Aussi, un microaimant crée un gradient de champ magnétique dans toutes les directions afin de pouvoir manipuler chaque qubit individuellement. Par contre, ces deux derniers éléments ne sont pas visibles sur la figure. b) Vue de côté du dispositif. La coupe est faite selon la ligne pointillée dans a). Les grilles bleues servent à ajuster les barrières tunnel entre les boîtes quantiques, alors que les grilles rouges permettent d'ajuster le potentiel chimique à l'intérieur des boîtes quantiques. De plus, la grille jaune est utilisée pour envoyer les impulsions RF utilisées pour faire bouger les électrons dans le gradient de champ magnétique et ainsi générer les rotations de spin. Pour simplifier le dispositif, il est supposé qu'il n'y a des électrons que dans la zone grise pâle où les grilles y créent un profil de potentiel 1D. c) Diagramme de stabilité schématisé d'une double boîte quantique. Les déplacements de 1 à 7 dans le diagramme permettent d'initialiser les qubits, de faire des opérations logiques et finalement de lire le résultat du calcul. Les traits plus minces dans le diagramme de stabilité indiquent qu'il faut une plus grande résolution en tension à ces étapes pour effectuer la lecture du spin. d) La tension de chaque grille est tracée et suit l'évolution du trajet dans le diagramme de stabilité. Les impulsions oranges sont appliquées sur la boîte quantique 1 (BQ1) et l'impulsion verte est appliquée sur la boîte quantique 2 (BQ2). e) L'état de spin des deux boîtes quantiques est affiché à chaque étape du calcul en suivant encore une fois la numérotation du trajet utilisée dans le diagramme de stabilité.

Chapitre 3

Caractéristiques techniques d'une plateforme de caractérisation rapide pour qubits de spin

Ce chapitre décrit les besoins des expériences avec des boîtes quantiques, ainsi que les caractéristiques techniques associées à ces expériences.

3.1 Besoins des mesures de diagrammes de stabilité

Selon le type de dispositif servant à créer des boîtes quantiques, la plage de tension nécessaire pour créer le confinement électrostatique et accumuler des électrons dans les boîtes quantiques peut varier. Par exemple, dans des hétérostructures de GaAs, les tensions utilisées sont relativement faibles et vont de -1 à 1 V [32]. À l'inverse, les dispositifs MOS (métal-oxyde-semiconducteur) peuvent être utilisés à des tensions allant jusqu'à $\pm$ 10 V en raison de la couche d'oxyde entre les grilles électrostatiques et la couche de semiconducteur. De plus, pour faire la lecture de spins, il faut un contrôle très précis des niveaux d'énergie des boîtes quantiques. Les besoins des expériences imposent donc les caractéristiques techniques des appareils utilisés [33]. Dans les tables 3.1 et 3.2 se trouve donc un résumé des besoins approximatifs liés aux mesures de diagrammes de stabilité. Il est à noter qu'en plus de ces exigences, il faut pouvoir programmer des impulsions arbitraires pouvant être synchronisées pour pouvoir tirer profit des grilles virtuelles. Plus de détails sur ce que sont les grilles virtuelles seront présentés au chapitre 6.

	Grilles	Source-drain	Spectroscopie	Mesure DC
Tension	$\pm$ 10 V	$<$ 10 mV	$\pm$ 10 mV	$\pm$ 1 V *
Résolution	100 μV	10 μV	10 μV	$<$ 14 bits **
Bande passante	10 Hz - 10 MHz	$<$ 1 MHz	100 MHz	10 - 100 MHz
Nombre de canaux	1 à 4 par BQ	1 par rangée	1 par BQ	1 par détecteur

TABLE 3.1 Caractéristiques techniques pour la caractérisation DC de boîtes quantiques. *Pour l'acquisition, des préamplificateurs sont utilisés, alors la valeur de tension peut être flexible. **Pas de besoin d'une haute résolution, encore une fois à cause des préamplificateurs.

	Mesure AC
Tension	$\pm$ 10 mV
Résolution	10 μV
Fréquence	100 Hz - 10 GHz
Nombre de canaux	Habituellement 1 par BQ

TABLE 3.2 Caractéristiques techniques pour la caractérisation AC de boîtes quantiques

3.2 Besoins pour la manipulation de spins

La manipulation cohérente de spins demande une grande précision des signaux de contrôle autant pour les signaux DC que AC. Il est particulièrement important de bien contrôler la phase et la durée des impulsions RF pour effectuer les portes logiques voulues. Les détails sur les exigences sont présentés dans la table 3.3. Selon l'expérience, il

DC / AC	Contrôle DC	Contrôle AC
Amplitude / Puissance	$>$ 10 mV	-20 dBm à 40 dBm *
Fréquence	-	1 - 50 GHz
Précision	1 μV	$<$ 10 Hz, $<$ 3 deg
Durée d'impulsion	-	10 ns à 1 μs
Résolution temporelle	$<$ 1 ns **	1 - 100 ns ***

TABLE 3.3 Caractéristiques techniques pour la manipulation de spins. *Pour atteindre, -40 dBm à basse température. Le but est d'atteindre au niveau de l'échantillon un bruit en dessous de $0.05\,\mathrm{nV_{RMS}/Hz^{1/2}}$ à 1 Hz. **Temps de montée et synchronisation entre impulsions DC et AC. ***Valeur typique, mais certaines opérations nécessitent une précision de 10 à 100 ps.

est nécessaire d'observer des déplacements d'électrons par effet tunnel très rapides ou de mesurer des temps de cohérence longs. Il faut donc avoir une fréquence d'échantillonnage allant jusqu'à 1 Géch./s ou une fenêtre d'échantillonnage de une ou deux secondes selon le cas. De plus, une grande flexibilité des paramètres des impulsions est nécessaire pour accommoder les différentes expériences. Ainsi, une séquence d'impulsions doit pouvoir

être programmée. Pour effectuer de la correction d'erreur, le délai entre la mesure de parité et l'application de la correction doit se faire en moins de $10\,\mu s$. Aussi, la dérive de l'horloge interne de l'appareil devrait être assez stable pour minimiser les erreurs de synchronisation lors de longues séquences d'impulsion. En règle générale, réduire la dérive pour atteindre le dixième de la plus petite impulsion ou du plus petit délai d'attente devrait être suffisant.

Chapitre 4

Plateforme Keysight

Keysight Technologies est un joueur majeur dans l'industrie de l'instrumentation scientifique et un partenaire de l'Institut quantique. Leurs appareils sont utilisés pour un grand nombre d'applications, mais ne sont pas parfaitement adaptés aux expériences avec les qubits de spin. Ils collaborent donc avec le groupe du Pr Michel Pioro-Ladrière pour améliorer leur solution existante pour l'information quantique, soit le «Quantum Engineering Toolkit» (QET). Le QET est composé de plusieurs instruments conçus pour travailler ensemble et de logiciels qui permettent de contrôler ces instruments. Les différentes composantes seront décrites dans les prochaines sections.

4.1 Instruments

Au cœur de la plateforme se trouve le châssis PXIe. PXI veut dire «PCI eXtensions for Instrumentation» en anglais et est un standard informatique permettant de rendre un système modulaire et flexible. PXIe est seulement une mise à jour pour suivre le standard PCI express (Peripheral Component Interconnect express) ou PCIe utilisé dans les ordinateurs. Comme son nom l'indique, le standard PXIe permet d'avoir dans un système plusieurs connexions PCIe entre un ordinateur et des instruments de mesure. Dans le cas du châssis, il y a 18 connexions disponibles et il peut être configuré selon les besoins des expériences. Un autre avantage majeur du standard PXIe est qu'il permet de synchroniser tous les instruments du châssis. Il est donc facile d'avoir une synergie entre les différents instruments. La synchronisation est également une composante importante en vue de l'utilisation de codes de correction d'erreurs quantiques. Lorsque le châssis est connecté à un ordinateur via une connexion PCIe, il est possible de contrôler tous les appareils présents sur le châssis

et de transférer des données jusqu'à 8 Go/s. Par contre, il n'est pas possible de débrancher le châssis ou de le redémarrer sans éteindre l'ordinateur. Sans quoi, l'ordinateur provoquera une panne de protection générale. Cette restriction n'est pas présente avec les périphériques USB par exemple.

Pour les mesures de caractérisation de boîtes quantiques, les deux appareils utilisés sont le générateur de signaux arbitraires («Arbitrary Waveform Generator» ou AWG en anglais) et le digitaliseur ou «digitizer» en anglais. Les caractéristiques techniques DC et AC de ces derniers sont donnés dans les tables 4.1 et 4.2 respectivement. Un générateur de signaux arbitraires permet de générer n'importe quelle onde programmée dans la mémoire de l'appareil. Cela permet l'utilisation de signaux plus complexes que l'onde sinusoïdale ou carrée qui viennent par défaut avec les instruments.

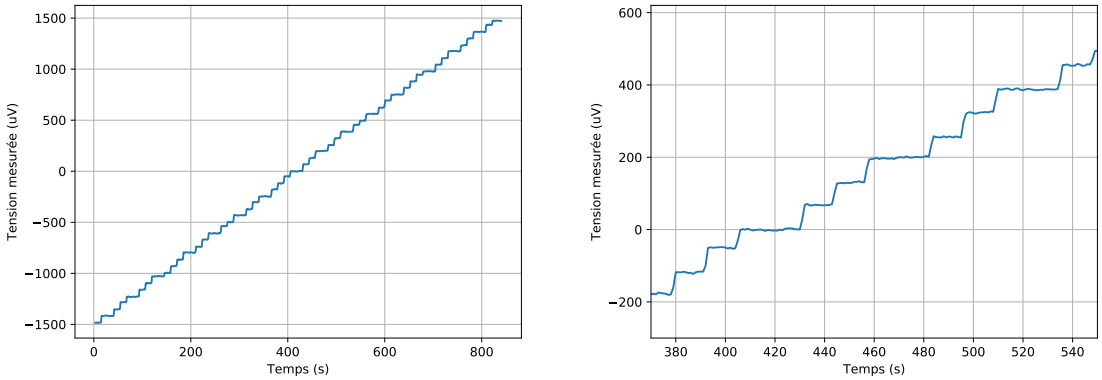
	Générateurs de signaux		Digitaliseurs	
	M3201A	M3202A	M3100A	M3102A
Plage de tension	±1.5 V	±1.5 V	6 Vpp (50 Ω) 20 Vpp (1 MΩ)	8 Vpp (50 Ω) 16 Vpp (1 MΩ)
Résolution	16 bits 45.8 µV	14 bits 183.1 µV	14 bits (10.8*) 1.1 mV	14 bits (10.6*) 1.3 mV
Bande passante (MHz)	200	400	100	200
Fréquence d'échantillonnage (Méch./s)	500	1000	100	500

TABLE 4.1 Caractéristiques techniques DC du QET. *Nombre effectif de bits compte tenu du bruit de l'appareil. La résolution en volt est calculée pour un «fullscale» de 2 (4 Vpp) pour couvrir le 3 Vpp du générateur.

	Générateurs de signaux		Source micro-ondes
	M3201A	M3202A	M9347A
Fréquence	DC - 200 MHz	DC - 400 MHz	50 MHz - 12 GHz
Précision (fréq.)	5.7 µHz	11.4 µHz	40 fHz
Précision (phase)	21.5 µdeg		$\approx 84 \times 10^{-9}$ deg

TABLE 4.2 Caractéristiques techniques AC du QET

Le générateur de signaux utilisé ici (M3201A) possède théoriquement une précision de 16 bits équivalant à 45.8 µV pour son intervalle de tension complet de -1.5 à 1.5 V. Pour vérifier la résolution, l'intervalle de -1.5 mV à 1.5 mV a été balayé avec le plus petit incrément de tension possible et mesuré avec un voltmètre. La tension est variée approximativement à chaque dix secondes pour permettre d'avoir plusieurs lectures du voltmètre pour chaque



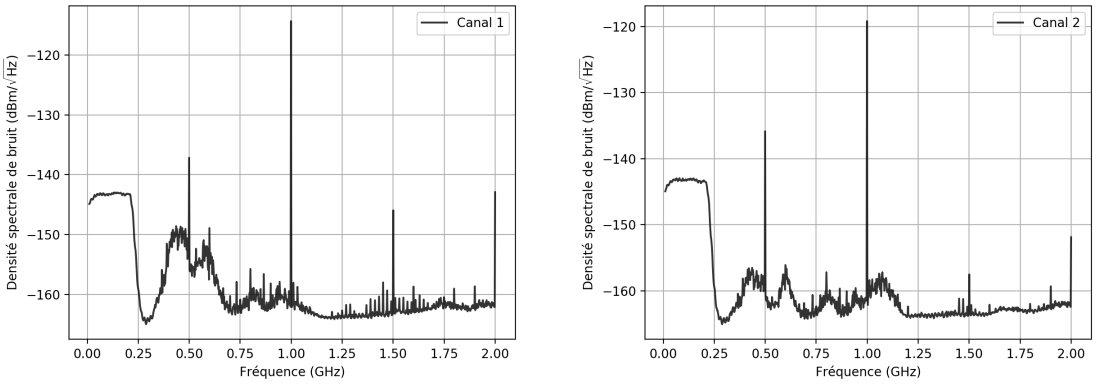
(**a**) Balayage de -1.5 à 1.5 mV avec incréments de 1 bit $(45.8\,\mu\mathrm{V}$ en théorie). La tension est variée à chaque dix secondes environ.

(**b**) Agrandissement de la figure de gauche autour de zéro. On voit qu'un incrément sur quatre est ignoré. De plus, les sauts de tension varient plutôt entre 56 et 70 $\mu\mathrm{V}$ d'un incrément à l'autre.

FIGURE 4.1 Résolution du premier canal du générateur de signaux (M3100A) dans la fente 8 du châssis

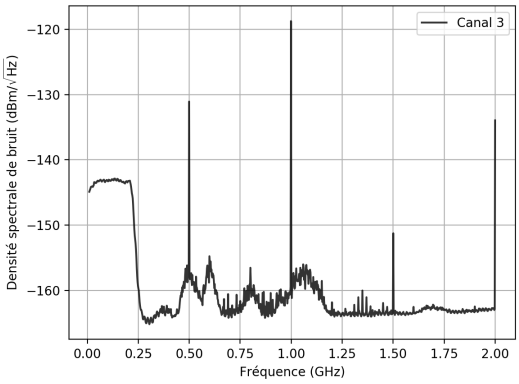
incrément du générateur de signaux. Cependant, comme on peut le voir à la figure 4.1, la résolution réelle est plus près de $70\,\mu\mathrm{V}$. De plus, certains incréments de tension sont sautés, ce qui, dans le pire cas, diminue la résolution de moitié à une valeur d'environ $140\,\mu\mathrm{V}$. La même situation se reproduit avec le digitaliseur utilisé (M3100A). La précision théorique de 14 bits, équivalant à $122.1\,\mu\mathrm{V}$ si un intervalle de tension de -1 à 1 V est utilisé, n'est pas réellement atteinte. Dans la documentation, on mentionne un nombre effectif de bits de 10.8, équivalant à 1.3 mV, pour tenir compte du bruit à l'intérieur de l'appareil. Par contre, le niveau de bruit des AWGs respecte la valeur attendue selon les caractéristiques techniques et vaut autour de $-145\,\mathrm{dBm/Hz}^{1/2}$ ou moins sur la plage de la bande-passante de l'appareil. Le niveau de bruit a été vérifié entre 10 MHz et 2 GHz avec un analyseur de signaux pour les quatre canaux du générateur de signaux (M3201A) de la fente 8 du châssis comme on peut le voir à la figure 4.2. Il est cependant important d'utiliser des filtres passe-bas avec les AWGs puisque le bruit est plus élevé à certaines fréquences comme à 1 GHz.

Il est important de mentionner que chaque instrument contient de la mémoire vive et un FPGA («Field-programmable gate array» en anglais) qui est un processeur spécialisé où il est possible de programmer chaque transistor pour faire le circuit le plus optimal afin de réaliser la tâche voulue. Il est donc possible d'envoyer une série d'instructions, voire un programme complet, à un ou plusieurs instruments et de l'exécuter sans devoir communiquer avec l'ordinateur. C'est ce qui rend la plateforme aussi flexible et performante.

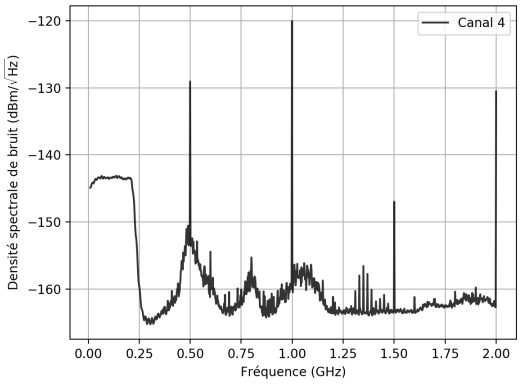


(**a**) Canal 1

(**b**) Canal 2



(**c**) Canal 3



(**d**) Canal 4

FIGURE 4.2 Densité spectrale de bruit pour les quatre canaux du générateur de signaux (M3201A) dans la fente 8 du châssis

4.1.1 Comparaison des besoins des mesures avec qubits de spin avec les caractéristiques techniques du QET

Maintenant que les caractéristiques techniques du QET sont connues, il est possible de les comparer aux besoins des expériences avec des qubits de spin. Dans sa présente version, le QET répond déjà à plusieurs des besoins, mais pas à tous. En ce qui concerne les mesures DC (voir la table 4.1), les besoins sont presque tous respectés. Il faudrait idéalement pouvoir atteindre des tensions plus élevées, mais des amplificateurs pourraient régler ce problème. Il faudrait seulement avoir une meilleure résolution sur les AWGs pour pouvoir précisément ajuster la tension source-drain des dispositifs et pour faire de la spectroscopie des niveaux d'énergie dans les boîtes quantiques. Pour les mesures AC (voir la table 4.2), la résolution est également insuffisante. Ainsi, l'utilisation d'atténuateurs est nécessaire. Les autres besoins sont déjà respectés avec les présentes caractéristiques techniques de l'appareil. Grâce à la modularité du QET, le nombre de canaux nécessaires pour contrôler un processeur quantique ne sera pas un problème.

Les besoins les plus stricts viennent de la manipulation cohérente de qubits (voir la table 3.3). La précision des tensions DC doit être améliorée, mais la précision des impulsions RF dépasse les besoins. Encore une fois, l'utilisation d'atténuateurs et d'amplificateurs pourrait être nécessaire pour atteindre la précision DC désirée et pour augmenter l'amplitude des signaux RF pour les expériences demandant une plus grande amplitude. De plus, le QET est limité à une fréquence de 12 GHz seulement, ce qui ne permet pas de faire toutes les expériences de qubits de spin qui peuvent aller jusqu'à 50 GHz. Un autre aspect important à considérer est que les impulsions pour les différentes portes logiques peuvent être assez courtes. Les impulsions de 10 ns peuvent théoriquement être faites par l'AWG M3202A grâce à sa fréquence d'échantillonnage de 1 Géch./s, mais une résolution maximale de 1 ns pourrait être atteinte. Dans la réalité, le délai peut être plus élevé, surtout si une synchronisation doit être faite. La synchronisation entre deux instruments peut prendre plusieurs centaines de nanosecondes. Ce délai supplémentaire rendrait la génération des impulsions les plus courtes impossible. Pire encore, les portes à deux qubits peuvent nécessiter une précision temporelle de l'ordre de 10-100 ps. Ce ne sont donc pas toutes les portes logiques qui peuvent être faites avec le QET.

Le bruit des AWGs est déjà très faible à température pièce et, avec l'utilisation d'atténuateurs, devrait être acceptable une fois rendu au niveau de l'échantillon. Dans le cas des digitaliseurs, le bruit est plus élevé ce qui réduit le nombre effectif de bits de l'appareil. La résolution passe donc de 14 bits à environ 11 bits, ce qui équivaut à une résolution autour du millivolt. Des amplificateurs sont donc nécessaires à la sortie du montage pour amplifier

les signaux lus par les digitaliseurs.

Sachant que le QET a été conçu pour les expériences de qubits supraconducteurs, il est fort probable que la plateforme puisse offrir de bonnes performances avec les qubits de spin également si les nouveaux besoins sont tenus en compte.

4.2 Logiciels

Même s'il est possible de contrôler le générateur de signaux arbitraires et le digitaliseur avec des commandes en Python, C++ ou LabVIEW pour les fonctions de base, il est beaucoup plus avantageux d'utiliser les programmes qui tirent profit du FPGA. Programmer le FPGA demande plus de temps et d'efforts, mais améliore significativement le temps de réponse des instruments. Normalement, il faut un expert avec le langage VHDL pour programmer un FPGA. Par contre, Keysight offre des langages de programmation graphique, similaires à LabVIEW, pour simplifier cette tâche. Ces programmes sont HVI et FPGA flow.

4.2.1 HVI

HVI est un acronyme pour «Hard Virtual Instrument» et permet de créer des instruments virtuels. Le programme des instruments virtuels est exécuté par les FPGAs dans le châssis au lieu de l'ordinateur branché à ce dernier. Cela permet de réduire la latence liée à la communication entre l'ordinateur et les instruments et donc de faire des mesures et du traitement de signal en temps réel. Ce gain en temps coûte par contre de la flexibilité lors de la programmation. En effet, un programme HVI est généralement fait pour une et une seule configuration matérielle bien précise. Il est donc nécessaire d'avoir plusieurs versions d'un même programme pour changer les canaux utilisés par exemple. Un avantage de HVI, autre que le gain en temps d'exécution des programmes, est qu'il permet de synchroniser l'exécution des tâches de plusieurs générateurs de signaux et/ou digitaliseurs présents sur le châssis. La figure 4.3 est un extrait de programme pour illustrer le fonctionnement de HVI. Un code HVI est composé de blocs qui servent de fonctions et commence au bloc «Start» en haut du programme pour se terminer au bloc «End» en bas du programme. Il faut suivre les flèches, qui vont généralement vers le bas, pour suivre l'exécution du programme. Voici les différents types de blocs et leur fonction.

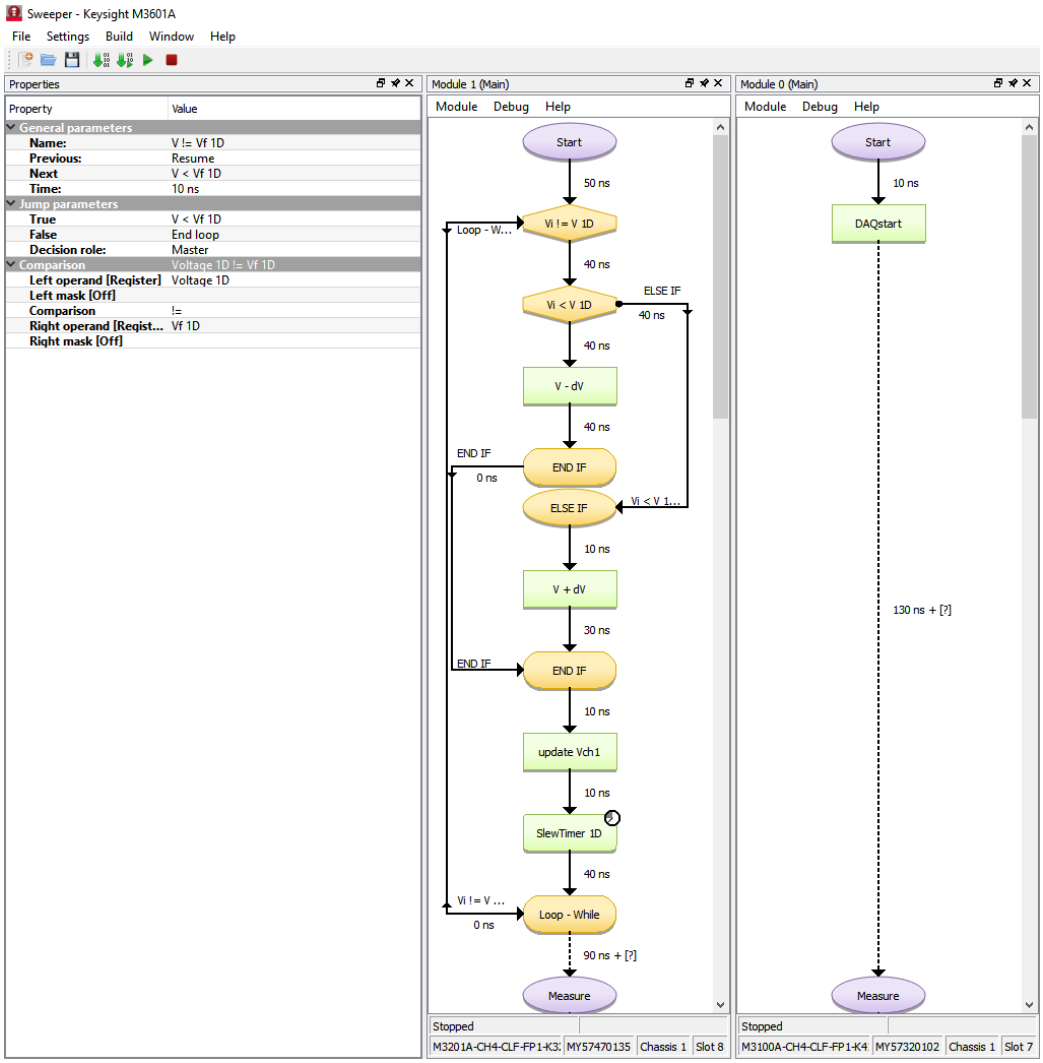


FIGURE 4.3 Extrait d'un programme HVI

— Blocs verts : Ces blocs servent à faire les fonctions de base. Ils servent, par exemple, à faire des opérations arithmétiques, lire ou écrire des valeurs en mémoire, changer les tensions des AWGs, envoyer un signal pour démarrer l'acquisition de mesure, attendre un certain temps avant de poursuivre l'exécution du programme, etc.

— Blocs jaunes : Les blocs jaunes servent, pour la plupart, à faire des opérations logiques. Par exemple, il est possible de faire des boucles FOR ou WHILE et des instructions conditionnelles avec le bloc SI. Il y a aussi un bloc jaune pour insérer un «subHVI» qui sont des sous-programmes HVI. Les subHVIs sont très utiles pour décomposer le code et ainsi le rendre plus facile à comprendre.

— Blocs mauves : Les blocs mauves servent à la synchronisation et il y en a deux prin-

cipaux. Les jonctions synchronisées forcent la synchronisation entre les différents appareils utilisés. Ainsi, le programme ne peut continuer que lorsque tous les appareils auront terminé l'exécution de leurs instructions respectives. L'instruction conditionnelle synchronisée, quant à elle, permet de vérifier une condition avant de poursuivre l'exécution du programme et d'aller à une jonction. Il est donc possible de sauter une partie du code ou de revenir en arrière. La synchronisation assure que tous les instruments reprennent l'exécution du programme au même point. Il est à noter que le bloc «Start» est un bloc mauve et peut être utilisé comme jonction.

L'exécution d'un programme HVI est déterministe en temps, ce qui veut dire que le temps d'exécution sera toujours le même, même si des paramètres sont changés. Par exemple, que la condition d'un bloc SI soit vraie ou fausse, le temps pris par le programme pour exécuter ce qui est contenu dans le bloc SI sera identique même si, disons, la condition fausse ne demande aucune action. Le programme ne fera qu'attendre un temps égal au temps d'exécution des instructions contenues dans la condition vraie. Ce déterminisme en temps a l'avantage de faire en sorte qu'il n'y ait pas de variabilité d'une exécution de programme à l'autre. Par contre, cet aspect peut parfois compliquer la programmation. Par exemple, il n'est pas possible d'avoir des jonctions synchronisées dans une boucle FOR ou WHILE. La boucle elle-même doit être synchronisée pour pouvoir utiliser d'autres blocs de synchronisation à l'intérieur. Il est donc nécessaire d'utiliser une condition et une jonction synchronisée pour faire une boucle synchronisée. Pour délimiter la boucle synchronisée, il faut utiliser une jonction synchronisée au début de la boucle et une condition synchronisée à la fin de la boucle. En reliant les deux, le code sera exécuté dans la boucle, puis la condition à la fin décidera si la boucle est répétée ou non. La seule boucle synchronisée possible est donc une boucle DO-WHILE. Une autre restriction concernant la synchronisation concerne les subHVIs. Si une jonction ou une condition synchronisée est utilisée à l'intérieur, le subHVI ne pourra pas être placé dans une boucle FOR ou WHILE ni dans un bloc SI. Il est donc plus facile de gérer la synchronisation entre instruments à l'extérieur des subHVIs.

Tel qu'il a été mentionné précédemment, la synchronisation se fait avec les blocs mauves. Cependant, une autre partie de la synchronisation, liée au temps d'exécution d'un programme, n'a pas été abordée. Chaque bloc nécessite un certain temps, typiquement entre 40 et 200 ns, pour être exécuté correctement. Si le temps alloué est trop court, il est possible que le bloc soit ignoré lors de l'exécution du programme. Malheureusement, ces temps doivent être entrés à la main et vérifiés constamment lorsque le programme doit être modifié. En effet, HVI essaie de garder constant le temps d'exécution du programme entre chaque jonction synchronisée. Alors, lorsqu'on ajoute un bloc au programme, HVI «vole»

du temps aux autres blocs du programme pour pouvoir ajouter le nouveau bloc. Cela a pour effet typiquement de ne pas donner assez de temps au nouveau bloc pour qu'il puisse être exécuté correctement, en plus de dérégler les temps d'autres blocs. Ce problème demande de corriger les erreurs à la main. Il faut donc se fier à la documentation pour connaître les contraintes de temps associées à chaque bloc. La documentation de Keysight s'avère donc cruciale pour quiconque voudrait utiliser HVI. Cependant, la documentation datant de 2017 n'est pas à jour et ne contient pas les contraintes de temps. Nous avons pu avoir une version plus récente de la documentation, pour les générateurs de signaux et digitaliseur, datant de 2018 grâce à notre collaboration. Il est donc fortement conseillé d'utiliser ces versions qui sont sauvegardées sur le serveur du département dans le répertoire du groupe.

Il a été mentionné plus haut qu'avec un bloc vert, il est possible de lire ou écrire des valeurs en mémoire. Plusieurs blocs peuvent interagir avec ces valeurs pour modifier le résultat du programme. Plus précisément, ces valeurs sont appelées des registres. Les registres peuvent être vus comme des variables que l'utilisateur peut définir avant l'exécution d'un programme HVI pour en changer le résultat final. Les registres peuvent aussi servir de variable temporaire dans le programme pour permettre de vérifier une condition dans une boucle WHILE par exemple. Il existe aussi les constantes qui ont un usage similaire, cependant, comme leur nom l'indique, elles ne peuvent pas être modifiées lors de l'exécution du programme. Les valeurs des constantes doivent être définies avant la compilation du programme. Une autre différence entre registres et constantes est qu'il y a seulement 16 registres de disponibles par instrument, alors qu'il n'y a pas de limite pour les constantes. Il est à noter aussi que les registres et constantes sont des entiers 32 bits. Il est possible de définir les registres et constantes à l'aide d'un nombre décimal, mais il faut spécifier quel est leur type parmi tension, fréquence, phase ou temps pour qu'une conversion soit effectuée. Il est cependant conseillé de vérifier que les conversions donnent le résultat attendu. Une erreur a été trouvée en utilisant un registre de temps avec un bloc «wait». Le résultat donnait un temps d'attente 10 fois trop long. L'erreur vient probablement du fait que le FPGA a une fréquence d'horloge de 100 MHz. Cela veut dire que la plus petite unité de temps est de 10 ns, alors que la conversion utilise probablement une unité de temps de 1 ns. Sinon, sachant les conversions, il est aussi possible de travailler seulement avec des entiers pour éviter les mauvaises surprises. Par exemple, si l'on ajoute un à la tension de sortie d'un générateur se signaux, cela correspond au plus petit incrément de tension que la précision de l'appareil permet. Il s'agit ici de $45.8\,\mu\mathrm{V}$.

Cette section a couvert les principaux aspects de HVI, mais il y a plusieurs détails superflus qui ont été laissés de côté. Il est donc conseillé de consulter la documentation pour de plus amples renseignements ou tout simplement d'ouvrir HVI pour se familiariser avec

l'interface.

4.2.2 FPGA flow

FPGA flow est un logiciel qui permet aussi de programmer un FPGA, mais est de plus bas niveau comparé à HVI et ne permet pas de communiquer entre plusieurs instruments. Avec HVI, il est possible de travailler avec des nombres à virgules et le logiciel s'occupe de faire les conversions pour passer, par exemple, de volts à un nombre entier. Par contre, avec FPGA flow, tout fonctionne avec des nombres entiers encodés typiquement dans 16 à 32 bits. Il faut donc s'assurer soi-même que les conversions sont faites correctement. Étant donné qu'il n'est pas possible d'utiliser des conditions comme le SI ni de faire des boucles et qu'il est difficile de modifier des données en mémoire, FPGA flow est surtout utile pour faire du traitement de signal en temps réel. La figure 4.4 montre un exemple de programme pour illustrer la structure d'un programme FPGA flow. Voici les différents éléments qui compose

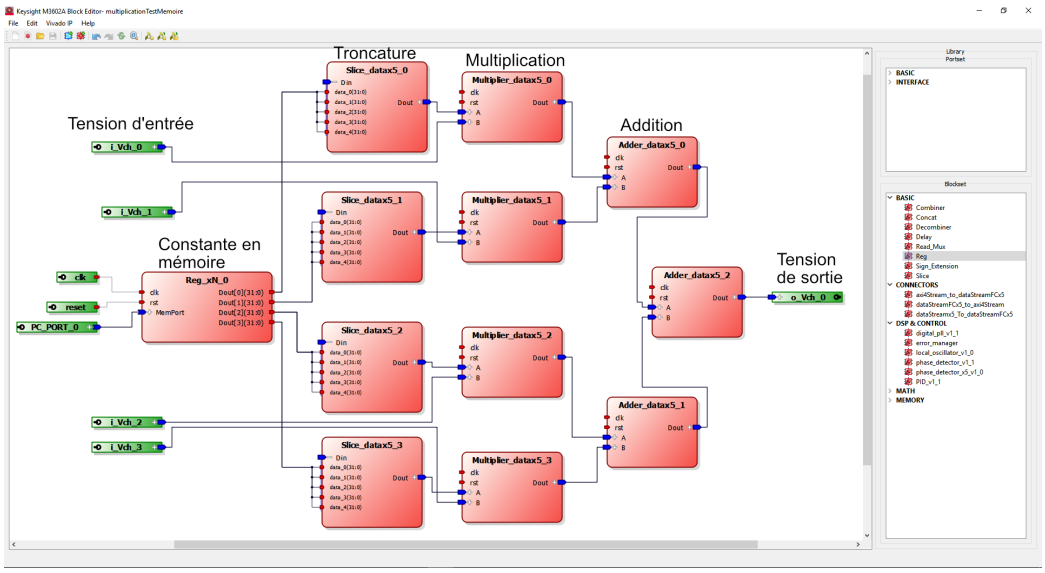


FIGURE 4.4 Extrait d'un programme FPGA flow. Ce programme multiplie chaque entrée d'un générateur de signaux à une constante, puis additionne le résultat de chaque canal. Ce résultat correspond au signal généré à la sortie du canal 1. Il s'agit seulement d'un exemple simple, mais qui est à la base de la multiplication matricielle dans un FPGA. Les signaux «clock» et «reset» ne sont pas connectés à tous les blocs pour améliorer la lisibilité du programme, mais doivent normalement l'être.

généralement un programme FPGA flow.

— Blocs verts : Ces blocs sont les entrées et sorties du programme. Il existe plusieurs types d'entrées et sorties, mais il suffit habituellement de prendre le même type que les connexions utilisées dans le programme par défaut pour que tout fonctionne.

- Blocs rouges : Ces blocs servent à faire les opérations de base telles que les additions, multiplications, troncatures, etc.
- Blocs jaunes : Ces blocs sont les sous-programmes créés avec FPGA flow.
- Blocs bleus : Ces blocs sont les sous-programmes créés avec un logiciel tiers de programmation FPGA comme Vivado.
- Connecteurs rouges : Ces connecteurs sont des ports qui transmettent l'information d'un bloc à l'autre. Les connexions rouges ne transmettent qu'un seul signal.
- Connecteurs bleus : Ces connecteurs sont des interfaces avec lesquelles il est possible d'interagir. Par exemple, c'est à travers une interface mémoire qu'il est possible de lire ou écrire des données dans la mémoire du FPGA. Les interfaces peuvent transmettre plusieurs signaux à la fois. Une connexion bleue est composée de plusieurs connexions rouges. Pour cette raison, elles facilitent la programmation et rendent aussi les gros codes plus lisibles.

Un peu comme HVI, on peut suivre l'exécution du programme en suivant les lignes qui sortent des blocs verts à gauche pour se rendre aux blocs verts à droite. Par contre, il arrive souvent que deux blocs n'utilisent pas le même nombre de bits en entrée et/ou sortie. Il faut donc utiliser un bloc de troncature pour sélectionner les bits à transmettre. Par exemple, dans certains cas, il faut garder les premiers bits et, dans d'autres cas, il faut garder les derniers bits d'un nombre. Sans cette étape, il se peut que le programme FPGA fasse lui-même une troncature, mais qu'elle ne soit pas la bonne, ou il se peut aussi que le programme ne compile tout simplement pas. Un détail important à connaître, surtout pour la troncature, concerne l'ordre des bits. Dans FPGA flow, la notation $(15:0)$ fait référence à un entier de 16 bits où le premier bit est le plus significatif, donc celui qui a la plus grande valeur numérique. Finalement, pour communiquer avec le programme FPGA, il est possible d'utiliser un PCport pour envoyer des paramètres depuis l'ordinateur ou un HVIport pour envoyer des paramètres depuis un programme HVI. Lorsque le programme est complété, il doit être compilé sur un serveur. La compilation prend habituellement une vingtaine de minutes, ce qui peut rendre fastidieuse la recherche de bogues. Il arrive aussi qu'il y ait des erreurs de compilation même si le programme est bien écrit. Le simple fait de compiler deux ou trois fois de suite suffit à régler le problème.

4.3 Exemple de programme FPGA : Amplificateur à détection synchrone

Un exemple qui montre bien la flexibilité de la plateforme est l'implémentation d'un amplificateur à détection synchrone («lock-in amplifier» en anglais) à partir d'un générateur de signaux arbitraires et d'un digitaliseur. Un programme comme celui-ci est cependant trop complexe pour être fait avec HVI ou FPGA flow. Il est donc nécessaire de faire la programmation en VHDL. Cette tâche a été effectuée par Azfar Badaroudine, un étudiant au département de génie électrique, et Larissa Njejimana, alors étudiante au même département. Le programme final est contenu dans un bloc FPGA qui peut être utilisé avec FPGA flow. Le fonctionnement du bloc FPGA de l'amplificateur est illustré à la figure 4.5. Grâce à ce programme, il est possible de faire des mesures en détection synchrone sans avoir à acheter de l'équipement supplémentaire. Il reste que le plus gros avantage est la possibilité de synchroniser ces mesures avec d'autres opérations à l'aide de HVI. Pour tester le bon fonctionnement et la performance de l'amplificateur, plusieurs tests ont été effectués. Parmi ceux-ci, la mesure de la bande passante, ainsi que de la réserve dynamique de l'amplificateur sont présentés à la figure 4.6. La réserve dynamique représente la capacité de l'amplificateur à éliminer le bruit aux fréquences différentes de la fréquence de référence.

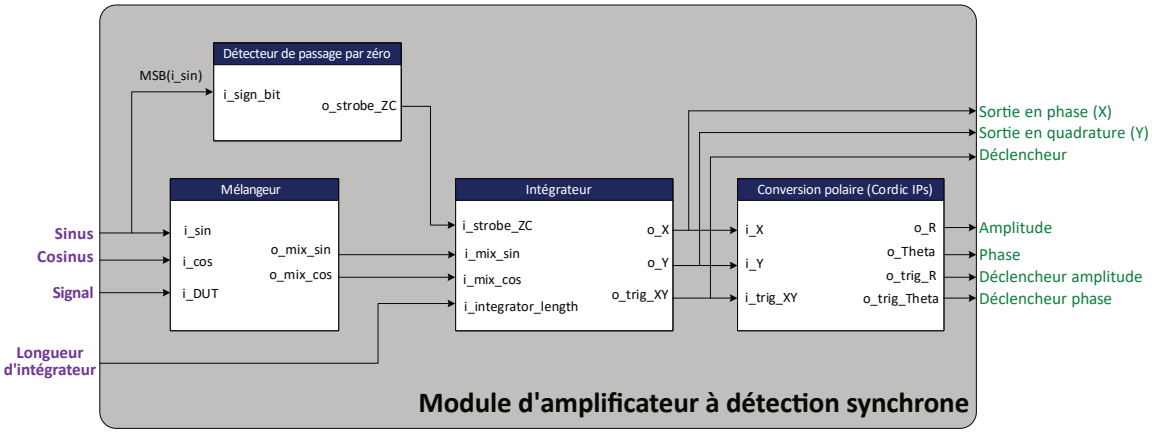
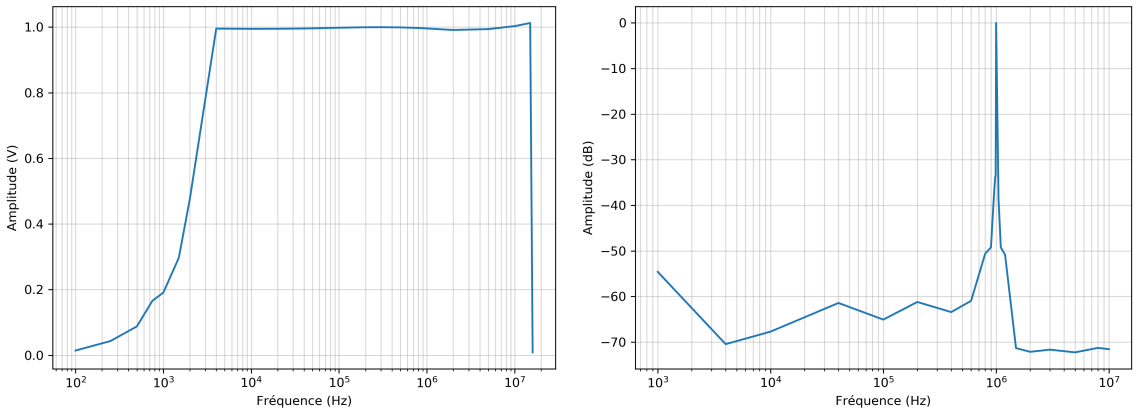


FIGURE 4.5 Schéma bloc de l'amplificateur à détection synchrone. Le module FPGA, installé dans le digitaliseur, prend en entrée les signaux sinus et cosinus générés à la fréquence de référence avec un générateur de signaux arbitraires, le signal à analyser, ainsi que la longueur d'intégrateur. Ce dernier paramètre représente le nombre de points à moyenner. À la sortie du module, le signal en phase et en quadrature et extrait. Le résultat peut être donné en coordonnées cartésiennes ou polaires. De plus amples détails sur le fonctionnement de chaque bloc, ainsi que la figure originale en anglais sont disponibles à la référence [34]. Note : la fonction MSB, acronyme anglais pour «most significant bit», renvoie le bit le plus significatif du sinus.



(**a**) Mesure de la bande passante

(**b**) Mesure de la réserve dynamique

FIGURE 4.6 Tests de performance de l'amplificateur à détection synchrone. Les mesures sont faites avec une longueur d'intégrateur de 1024. a) Un signal sinusoïdal de fréquence variable et d'amplitude de 1 V est mesuré avec l'amplificateur. La fréquence du signal et la fréquence de référence sont les mêmes. La plage de fréquence où il y a le moins d'atténuation est environ de 4 kHz à 15 MHz. b) Le même signal de fréquence variable qu'en a) est mesuré à une fréquence de référence de 1 MHz. La fréquence de référence est fixe pour ce test. L'amplitude du signal de 1V est bien mesurée à 1 MHz et l'amplitude du même signal décroît rapidement aux fréquences autour de la fréquence de référence.

Chapitre 5

Programmation de la plateforme pour les mesures en temps réel

5.1 Diagrammes de stabilité avec HVI

La programmation des diagrammes de stabilité est basée sur la génération de rampes de tension et de la synchronisation de ces rampes avec un digitaliseur pour l'acquisition de données. Les différentes étapes d'une rampe sont montrées dans la figure 5.1. La fonction d'une rampe est de changer la tension initiale du générateur de signaux (V_i) pour atteindre la tension finale désirée (V_f) selon les paramètres spécifiés. Ces paramètres sont le taux de balayage, la hauteur de marche, l'incrément de tension (dV), le temps de stabilisation, le temps d'intégration et le temps de pause. Le taux de balayage est une valeur donnée en volts par seconde et dicte la vitesse à laquelle la tension est changée entre deux points de mesure. L'incrément de tension représente le plus petit changement de tension qu'il est possible d'appliquer et permet de calculer le nombre de points contenu dans une rampe. Dans un sens, l'incrément de tension dicte le taux de balayage le plus grand qu'il est possible d'obtenir puisque chaque changement de tension prend un certain temps. Ainsi, plus il y a de points dans une rampe, plus cette rampe sera longue à générer. La hauteur de marche donne le changement de tension effectué entre deux points de mesure et est calculée à partir du nombre de points demandé selon l'équation 5.1. Le taux de balayage et la hauteur de marche doivent être choisis en fonction de l'échantillon testé. Si ce dernier est très sensible aux grandes variations de tension, il est préférable de diminuer ces deux paramètres. Certains échantillons testés durant ce projet avaient des couches minces d'oxyde pouvant facilement être endommagées si une différence de potentiel trop grande était appliquée rapidement. D'autres échantillons avaient de grosses grilles électrostatiques

qui agissaient comme un condensateur. Il était donc préférable de faire un balayage en tension lent pour permettre à ce condensateur de se charger ou décharger lentement et ainsi éviter de générer de gros courants dans l'échantillon. Ces courants peuvent endommager l'échantillon ou fausser les données en ajoutant une contribution non volontaire au courant total circulant dans l'échantillon. Pour éviter cet effet, il est aussi possible d'utiliser un temps de stabilisation qui sert justement à attendre que l'échantillon redevienne dans un état stable après avoir changé la tension. Le temps d'intégration, quant à lui, doit être spécifié dans les paramètres pour permettre la synchronisation avec le digitaliseur. Ce temps est donné par $t_{intégration} = t_{mesure} \times$ Nb de pts à moyenner. Le temps de pause quant à lui peut servir de 2^{e} temps de stabilisation ou de temps pour ajuster les paramètres d'autres appareils lors de la mesure.

$$\text{Hauteur de marche} = \frac{|V_f - V_i|}{\text{Nb de pts} - 1} \tag{5.1}$$

Un diagramme de stabilité est composé d'une succession de rampes, mais il y a plusieurs

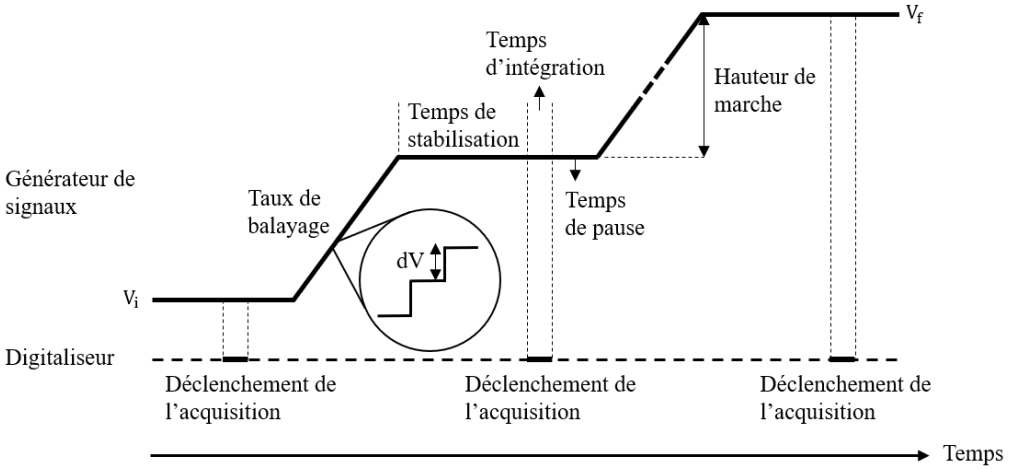


FIGURE 5.1 Visualisation du signal émis par le générateur de signaux arbitraires lors d'une rampe de tension (en haut) et des acquisitions par le digitaliseur (en bas). L'objectif d'une rampe est de se rendre à la valeur de tension finale V_f à partir de la tension initiale V_i en respectant la vitesse de balayage entre chaque mesure, le temps d'intégration lors de chaque mesure, ainsi que les autres paramètres spécifiés dans la figure. Les lignes pointillées entre le signal du générateur de signaux et les acquisitions du digitaliseur montrent là où les deux instruments doivent être synchronisés. Merci à Marc-Antoine Genest de m'avoir permis de réutiliser et modifier sa figure.

méthodes pour faire une rampe de tension basées sur différentes boucles. Une première méthode utilisant une boucle FOR a été implémentée par Marc-Antoine Genest à l'automne 2018. Par contre, HVI étant un programme toujours en développement, il y avait des problèmes avec les boucles et la synchronisation à ce moment-là. Ainsi, il n'était pas possible de mettre des temps d'attente dans une boucle sans perdre la synchronisation. Il était donc

nécessaire de synchroniser les opérations sur le générateur de signaux arbitraires et le digitaliseur manuellement pour obtenir le taux de balayage voulu. D'où l'idée d'utiliser une boucle FOR et de fixer à l'avance la plupart des paramètres, tels que le taux de balayage et le nombre de points mesurés, pour rendre la synchronisation simple à faire. Cette implémentation empêchait d'avoir la flexibilité de changer les paramètres de rampe facilement. Il était donc nécessaire d'avoir un programme HVI différent pour chaque rampe différente. Étant donné les problèmes rencontrés avec HVI, un programme FPGA en VHDL a été développé par Larissa Njejimana pour rendre la génération de rampes plus flexible. Un rapport a été rédigé par Larissa pour expliquer le fonctionnement du programme [35]. Avec ce programme, l'utilisateur pouvait choisir les paramètres de la rampe et la synchronisation était gérée par l'ordinateur avec un protocole de type «poignée de main». Ainsi, lorsque le générateur de signaux a atteint la tension désirée et qu'une mesure est nécessaire, il envoie un signal à l'ordinateur qui, à son tour, envoie un signal au digitaliseur pour que la mesure soit faite. Le digitaliseur doit ensuite dire à l'ordinateur quand la mesure est terminée pour que le générateur puisse continuer la rampe. Bien que fonctionnelle, cette méthode a le désavantage de ralentir considérablement les instruments puisqu'il faut attendre que l'ordinateur échange les signaux de synchronisation. Puisqu'une communication avec l'ordinateur prend un temps de l'ordre de 1 ms, il est préférable de retirer l'ordinateur de l'équation et de faire la synchronisation directement dans le châssis. Ce travail a été effectué avec HVI, mais a nécessité l'utilisation de plusieurs registres pour sauvegarder le progrès de la rampe et gérer les poignées de main. C'est cette version du programme, combinant le programme FPGA en VHDL et HVI pour la synchronisation, qui a été utilisée pour obtenir les résultats qui seront présentés dans les prochaines sections.

Toutefois, les problèmes de synchronisation avec les boucles ont été réglés après une mise à jour de HVI en hiver 2019. Ainsi, il est devenu possible de synchroniser les mesures avec les rampes automatiquement. Ce qui veut dire qu'il n'était plus nécessaire de fixer de paramètres pour pouvoir faire manuellement la synchronisation. Suivant les recommandations de Nizar Messaoudi, ingénieur chez Keysight, la boucle synchronisée a été utilisée pour générer les rampes et déclencher les mesures au bon moment. Un autre avantage de cette technique est que le même programme peut être réutilisé pour différents paramètres de balayage sans avoir à ajuster les délais entre les blocs. Il suffit d'utiliser un temps d'attente dans la boucle pour ajuster le taux de balayage à la valeur désirée. Alors, avec les problèmes de synchronisation résolus, un programme HVI pour faire un diagramme de stabilité a été fait et est présenté à la figure A.2. Il est à noter que cette version du programme n'est pas encore fonctionnelle à 100% puisque certaines rampes sont répétées trop souvent et la synchronisation n'est pas parfaite. Il existe aussi une version simplifiée de ce programme qui fait seulement des rampes 1D et qui est présenté à la figure A.1. Cette version est fonctionnelle et a servi de base pour la

version faisant les diagrammes 2D. Pour revenir au programme qui mesure des diagrammes de stabilité, l'exécution générale du programme se déroule comme suit : pour éviter des sauts de tension abruptes, la première rampe d'un diagramme doit débuter à la valeur de tension appliquée par le générateur de signaux puis se rendre à V_i. La rampe est ensuite exécutée pour se rendre à la V_f. Cette séquence effectue une seule trace du diagramme, mais la séquence est répétée au sein d'une 2e rampe pour faire une rampe 2D. Cela permet de varier deux tensions en même temps et de faire un diagramme complet.

Telle qu'elle est décrite plus haut, la première étape de l'algorithme consiste à atteindre les tensions initiales du diagramme de stabilité à partir des tensions présentement appliquées sur les deux canaux du générateur de signaux. Même si cette opération semble simple à première vue, elle est plus compliquée qu'à la normale à cause de l'utilisation de nombres binaires par le FPGA. À la base, les nombres binaires permettent seulement d'avoir des entiers non signés. Les entiers non signés sont des entiers qui, comme leur nom l'indique, n'ont pas de signe. Il est tout de même nécessaire de pouvoir utiliser des nombres négatifs pour pouvoir appliquer des tensions négatives avec le générateur de signaux. Une méthode standard pour représenter les nombres entiers négatifs avec des nombres binaires est le complément à deux. Cette méthode permet de représenter les nombres de -2^{N-1} à $2^{N-1}-1$ où N est le nombre de bits utilisés pour représenter les nombres. Autrement dit, parmi tous les nombres pouvant être représentés avec N bits, la moitié est utilisée pour les nombres négatifs et l'autre moitié est utilisée pour les nombres positifs. Dans le cas du générateur de signaux avec une précision de 16 bits, il est possible de représenter $2^{16} = 65536$ valeurs. Alors, les nombres positifs sont encodés dans l'intervalle de 0 à 32 767 et les nombres négatifs entre 32 768 et 65 535. Étant donné que l'intervalle de tension du générateur de signaux est de -1.5 à 1.5 V, chaque entier peut être associé à une tension en utilisant le complément à deux comme dans le tableau 5.1. Même si le complément à deux a l'avantage d'utiliser

Tension	Entier	Complément à deux
1.5 V	32767	0111 1111 1111 1111
1 V	21844	0101 0101 0101 0100
0.5 V	10922	0010 1010 1010 1010
91.6 µV	2	0000 0000 0000 0010
45.8 µV	1	0000 0000 0000 0001
0 V	0	0000 0000 0000 0000
-45.8 µV	65535	1111 1111 1111 1111
-0.5 V	54614	1101 0101 0101 0110
-1 V	43692	1010 1010 1010 1100
-1.5 V	32769	1000 0000 0000 0001

TABLE 5.1 Conversion des nombres décimaux en nombres binaires avec le complément à deux

les mêmes algorithmes de calcul pour l'addition, la soustraction et la multiplication que pour les nombres non signés, il pose un problème quand il est utilisé avec HVI. HVI est à l'interface entre la programmation de bas et de haut niveau et mélange des concepts de chaque niveau de programmation. Par exemple, HVI peut gérer des nombres à virgule et faire les conversions appropriées avant d'envoyer les paramètres au FPGA. Par contre, lorsqu'il compare deux nombres, il utilise la représentation du complément à deux qui est de bas niveau avec un algorithme de comparaison de haut niveau qui ne tient pas compte de la représentation des nombres. Ainsi, -1 V (43 692) est plus grand que 1 V (21 844). Ceci pose évidemment un problème. Si les deux nombres comparés sont tous les deux positifs ou négatifs, il n'y a pas de correction à faire. Cependant, s'il faut comparer un nombre positif et négatif, la condition testée doit être inversée pour que les nombres négatifs soient considérés comme plus petits que les nombres positifs et non l'inverse. À partir de la tension appliquée sur le canal X du générateur de signaux, l'algorithme 1 se rapproche de la tension désirée d'un incrément de tension en respectant le taux de balayage spécifié. L'algorithme compare aussi adéquatement les nombres positifs et négatifs. Une fois que plusieurs incréments de tension sont effectués et que la tension désirée est atteinte, une mesure de tension est faite par le digitaliseur. Il est à noter qu'une mesure n'est pas effectuée à chaque incrément de tension, mais seulement lorsqu'un multiple de la hauteur de marche est atteint. L'algorithme 2 décrit la procédure de mesure faite à chaque incrément de tension. En un premier temps, le programme détermine s'il doit sauter ou non une mesure à l'incrément de tension actuel. Si une mesure doit être faite, un signal de déclenchement est envoyé pour faire la mesure en respectant les temps de stabilisation, d'intégration et de pause.

L'algorithme 3 utilise les algorithmes précédents pour faire un diagramme de stabilité complet. Il s'agit essentiellement de deux boucles imbriquées, une pour chaque tension à varier, où, à chaque itération, les deux tensions sont ajustées pour commencer aux tensions initiales du diagramme et aller vers les tensions finales. L'algorithme décide selon les paramètres de départ après combien d'incréments de tension une mesure doit être faite. Le nombre d'incréments à faire pour chaque marche est déterminé par l'équation 5.2. L'envoi des paramètres, la lecture des données ainsi que le départ et l'arrêt du programme sont faits à l'aide de commandes en Python à partir de l'ordinateur connecté au châssis.

$$\text{Nombre d'incréments par marche} = \left\lfloor \frac{\text{Hauteur de marche}}{\text{dV}} \right\rfloor - 1 \tag{5.2}$$

Données :
V_i/V_f : Tension initiale/finale à atteindre
dV : Amplitude des incréments de tension ($45.8\,\mu V$ par défaut)
$T_{balayage}$: Temps de balayage $\left(\frac{dV}{\text{Taux de balayage}} - T_{\text{exéc. HVI}} \text{ ou } 0 \text{ si négatif}\right)$
`/* Début de l'algorithme */`
si $V_i/V_f < V_{canal}$ *X* **alors**
  utiliser des incréments négatifs pour la tension ;
sinon
  utiliser des incréments positifs pour la tension ;
fin
Définir la hauteur des marches selon le paramètre dV et le signe des incréments ;
Initialiser à 0 le compteur d'arguments (V_i/V_f ou V_{canal} X) négatifs ;
si V_i/V_f *est négatif* **alors**
  incrémenter le compteur d'arguments négatifs
fin
si V_{canal} *X est négatif* **alors**
  incrémenter le compteur d'arguments négatifs
fin
si *le compteur d'arguments négatifs égale 1* **alors**
  inverser le signe de la hauteur des marches ;
  calculer la nouvelle tension en ajoutant la hauteur des marches à la tension V_{canal} X ;
sinon
  calculer la nouvelle tension en ajoutant la hauteur des marches à la tension V_{canal} X ;
fin
Appliquer la nouvelle tension sur le canal X ;
Attendre $T_{balayage}$ pour respecter le taux de balayage ;

Algorithme 1 : Aller à V_i/V_f (canal X)

Données :
i : compteur d'incréments 1D
Nombre d'incréments par marche 1D
Temps de stabilisation
Temps d'intégration + temps de pause
`/* Début de l'algorithme */`
si *i < nombre d'incréments par marche 1D* **alors**
  i = i + 1 ;
sinon
  Attendre le temps de stabilisation ;
  Déclenchement de la mesure du canal X ;
  Attendre le temps d'intégration et de pause ;
  i = 0 ;
fin

Algorithme 2 : Mesurer V_{canal} X

Données :
V_i 1D, V_f 1D, V_i 2D, V_f 2D
Compteur d'incréments 1D, Compteur d'incréments 2D
Nombre d'incréments par marche 2D
Temps de stabilisation, Temps d'intégration
`/* Début de l'algorithme */`
tant que V_i 2D *n'égale pas* V_{canal} *2D* **faire**
  Exécuter le subHVI «Aller à V_i canal 2D» ;
fin
tant que V_i 1D *n'égale pas* V_{canal} *1D* **faire**
  Exécuter le subHVI «Aller à V_i canal 1D» ;
fin
Démarrer l'acquisition en mode attente d'un signal de déclenchement ;
Attendre le temps de stabilisation ;
Déclenchement de la mesure des canaux 1D et 2D ;
Attendre le temps d'intégration et de pause ;
Initialisation du compteur d'incréments 1D à 0 ($i = 0$) ;
début Boucle 1D
  Exécuter le subHVI «Aller à V_f canal 1D» ;
  Exécuter le subHVI «Mesurer V_{canal} 1D» ;
  si V_{canal} *1D n'égale pas* V_f 1D **alors**
    retourner au début de la boucle ;
  fin
fin
début Rampe 2D
  Initialisation du compteur d'incréments 2D à 0 ($j = 0$) ;
  début
    Exécuter le subHVI «Aller à V_f canal 2D» ;
    $j = j + 1$;
    si *j est plus petit que le nombre d'incréments par marche 2D* **alors**
      retourner au début de la boucle ;
    fin
  fin
  tant que V_i 1D *n'égale pas* V_{canal} *1D* **faire**
    Exécuter le subHVI «Aller à V_i canal 1D» ;
  fin
  Attendre le temps de stabilisation ;
  Déclenchement de la mesure des canaux 1D et 2D ;
  Attendre le temps d'intégration et de pause ;
  $i = 0$;
  début Boucle 1D
    Exécuter le subHVI «Aller à V_f canal 1D» ;
    Exécuter le subHVI «Mesurer V_{canal} 1D» ;
    si V_{canal} *1D n'égale pas* V_f 1D **alors**
      retourner au début de la boucle 1D ;
    fin
  fin
  si V_{canal} *2D n'égale pas* V_f 2D **alors**
    retourner au début de la rampe 2D ;
  fin
fin

Algorithme 3 : Mesure d'un diagramme de stabilité

5.2 Mesures en temps réel

L'implémentation des mesures de diagrammes de stabilité avec un FPGA permet de gagner beaucoup de temps comparé à un système où la logique des mesures et contrôlée par ordinateur. La figure 5.2 montre une comparaison de la vitesse d'acquisition d'un diagramme de stabilité par un SMU et un FPGA. Un SMU, ou «source measure unit» en anglais, est un appareil de mesure pouvant appliquer ou mesurer soit des tensions ou des courants. Par contre, un SMU n'est pas programmable et doit communiquer avec un ordinateur à chaque fois qu'une mesure ou lecture est nécessaire. Dans la figure 5.2, on voit les signaux générés par le FPGA et le SMU pour les deux grilles utilisées pour faire le diagramme. En regardant le temps nécessaire pour compléter la séquence, il est possible de constater que le FPGA termine le diagramme en environ 3.5 millisecondes, soit plus de 1000 fois plus rapidement que le SMU. Ce temps est assez petit pour permettre des mesures en temps réel. Ainsi, il est possible de mesurer un diagramme si rapidement qu'il peut être répété plusieurs fois par seconde et fournir des informations sur ce qui se passe avec un échantillon à travers un flux vidéo en temps réel. C'est pourquoi ce type de mesure est aussi appelé «mode vidéo» [29]. Par contre, dans le cas où il faut sauvegarder la vidéo, il faut prévoir plus de temps pour que les données soient enregistrées avant de faire la prochaine mesure. La mesure de la figure 5.2 a été effectuée sans enregistrement. Si un enregistrement est effectué, le temps nécessaire pour mesurer et enregistrer un diagramme augmente à 30 ms si les données sont sauvegardées avec la bibliothèque Numpy de Python ou 80 ms si Python est utilisé seul. Ces temps donnent un taux de rafraichissement d'image de 33.3 et 12.5 images par seconde (IPS) respectivement. Pour comparaison, le taux de rafraichissement d'image des postes de télévision est d'environ 30 IPS et le cerveau humain perçoit du mouvement avec des images affichées à environ 15 IPS ou plus [36]. Ce nombre constitue donc une cible à atteindre pour dire qu'une mesure se fait réellement en mode vidéo et en temps réel. Bien entendu, plus le diagramme est gros, plus il faut de temps pour le mesurer. Ainsi, pour que les mesures soient assez rapides pour paraître fluides, il faut limiter la plage de tension couverte et le nombre de points mesurés. Par exemple, la mesure de la figure 5.2 couvre une plage de 30 mV par 150 mV avec $31 \times 31 = 961$ mesures. À la figure 5.3, deux diagrammes de stabilité complets sont présentés pour comparer la précision entre le SMU et le FPGA. Comme on peut le voir, le FPGA reproduit bien la mesure du SMU pour un temps d'intégration identique. Ces mesures ont été prises à température pièce sur un transistor pour tester la plateforme.

Une fois que les rampes de tensions sont programmées avec HVI et qu'il est possible de mesurer des diagrammes de stabilité rapidement, il devient intéressant d'améliorer les programmes HVI en y ajoutant des fonctionnalités plus avancées pour faciliter l'initialisation

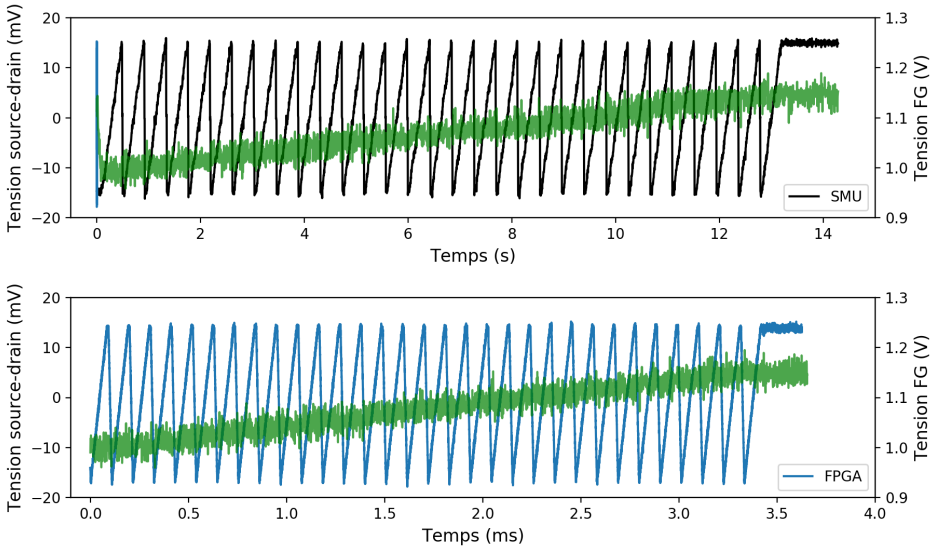
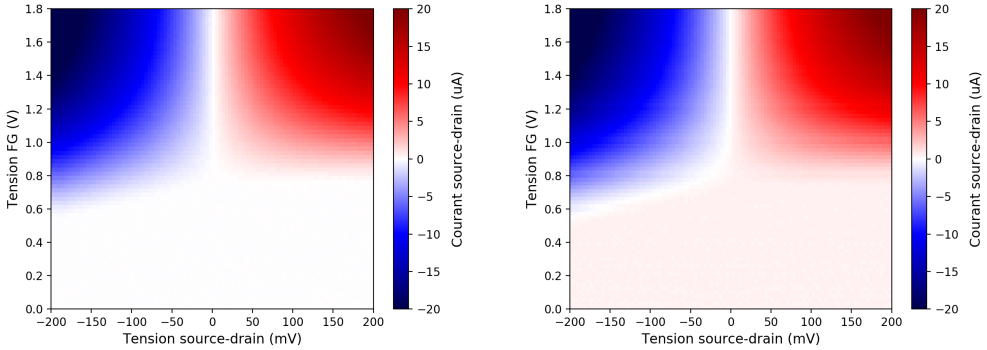


FIGURE 5.2 Comparaison du temps d'exécution d'un diagramme de stabilité entre un SMU et un FPGA. La tension est variée de -15 à 15 mV sur l'axe des x et de 1 à 1.15 V sur l'axe des y (31x31 points). Le taux de balayage est 2 V/s pour le SMU et de 1 kV/s pour le FPGA. Le temps d'intégration pour le SMU est de 80 µs et de 500 ns pour le FPGA. Les taux de balayage et les temps d'intégration correspondent approximativement aux meilleurs paramètres pour mesurer un diagramme de stabilité le plus rapidement possible en tenant compte des limitations de chaque appareil.



(**a**) Diagramme de stabilité mesuré avec un SMU

(**b**) Diagramme de stabilité mesuré avec un FPGA

FIGURE 5.3 Comparaison de la précision entre un SMU et FPGA pour un même diagramme de stabilité de 101 x 71 points. Un taux de balayage de 300 mV/s et un temps d'intégration de 80 us a été utilisé dans les deux mesures. Il est possible de voir un léger décalage de tension sur la courbe du FPGA, puisqu'il y a du courant à zéro tension. Il y a aussi des séparations visibles dans le diagramme du FPGA probablement dues au fait que le diagramme complet a été acquis en plusieurs mesures. Les conditions expérimentales ont donc pu varier légèrement entre les mesures.

de boîtes quantiques. Les grilles virtuelles sont une de ces fonctionnalités et seront décrites au prochain chapitre.

Chapitre 6

Grilles virtuelles

Un outil très puissant qui peut être utilisé avec les diagrammes de stabilité est le concept de grille virtuelle [37]. Lors de la conception d'une boîte quantique, chaque grille électrostatique a un rôle qui lui est attribué. Que ce soit de contrôler le potentiel chimique de la boîte quantique ou la largeur d'une barrière tunnel, une grille est utilisée pour ajuster son environnement électrostatique immédiat selon les besoins de l'expérience. Cependant, toutes les grilles d'un dispositif sont si proches l'une de l'autre que la tension d'une modifie l'environnement électrostatique des boîtes quantiques avoisinantes. Il devient alors rapidement difficile d'ajuster le potentiel de plusieurs grilles, pour créer plus d'une boîte quantique par exemple, car l'ajustement d'une grille affecte l'ajustement des autres grilles. Les grilles virtuelles règlent ce problème en éliminant le couplage capacitif entre les grilles. Les grilles peuvent alors être utilisées comme prévu lors de la conception du dispositif, ce qui facilite les manipulations lors des expériences et, par le fait même, l'analyse des résultats étant donné que l'expérience est «simplifiée» en quelque sorte. Un exemple concret où la mesure d'un diagramme de stabilité avec les grilles virtuelles s'avère utile est l'initialisation de boîtes quantiques. Puisque les lignes de transitions de charges deviennent droites lorsque le couplage entre les grilles est retiré, il devient plus simple de «naviguer» dans le diagramme de stabilité. C'est d'ailleurs cette astuce qui a été illustrée à la figure 2.9 c).

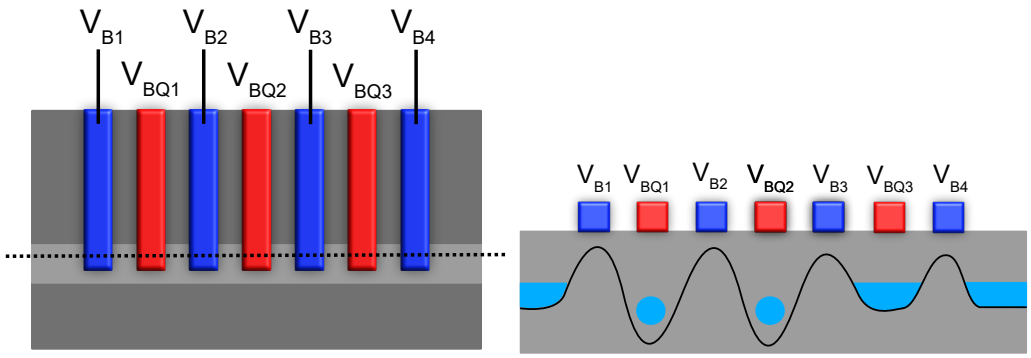
Pour utiliser les grilles virtuelles, il faut tout d'abord manuellement créer une double boîte quantique. Ensuite, l'influence des grilles autour des deux boîtes quantiques sur les niveaux d'énergie des boîtes quantiques est quantifiée. Cette influence se voit par un décalage des lignes de transition de charge dans le diagramme de stabilité lorsque la tension d'une grille est légèrement modifiée. L'influence (α) de chaque grille est mesurée en fonction du décalage des lignes de transition des boîtes quantiques (ΔL) et du changement de tension (δV) selon l'équation 6.1 [37].

$$\alpha_{ij} = \frac{\Delta L_i}{\delta V_j} \tag{6.1}$$

Chaque élément α est inséré dans une matrice de capacités croisées (CC) comme à l'équation 6.2. Cette matrice n'est qu'un cas particulier avec 5 grilles, mais le nombre de grilles peut évidemment être différent. L'équation 6.3 décrit de façon générale l'équation 6.2. Il est à noter que les éléments de matrice associés aux grilles servant à ajuster une barrière tunnel sont nuls, sauf sur la diagonale, puisque l'effet des grilles sur les barrières tunnel n'est pas compensé. L'inverse de cette matrice donne la correction à faire sur chaque grille du dispositif pour compenser la variation de tension d'une grille qui se fait balayer. L'équation 6.4 donne la tension à appliquer sur chaque grille physique (V_j) selon la tension voulue sur une ou plusieurs grilles virtuelles $\tilde{V}_i$.

6.1 Simulation de diagrammes de stabilité

Pour tester l'algorithme avant de l'implémenter dans un FPGA, un simulateur de boîtes quantiques a été utilisé. Ce simulateur a été programmé par Marc-Antoine Genest en utilisant le modèle de Thomas-Fermi [26]. Le dispositif simulé ici est une rangée unidimensionnelle de quatre grilles contrôlant le confinement, ou autrement dit les barrières tunnel, ($B1$, $B2$, $B3$ et $B4$) et de trois grilles contrôlant le potentiel chimique de chaque boîte quantique ($BQ1$, $BQ2$ et $BQ3$). Pour former une double boîte quantique, les grilles sont placées dans l'ordre suivant : $B1$, $BQ1$, $B2$, $BQ2$, $B3$. Les grilles $BQ3$ et $B4$ ne servent pas à former la double boîte quantique, mais plutôt à tester l'efficacité des grilles virtuelles lorsqu'une troisième boîte quantique se forme. Un schéma du dispositif simulé est présenté à la figure 6.1. Chaque grille de barrière est à une tension de -0.5 V pour créer le confinement. Les grilles $BQ1$ et $BQ2$ sont variées durant les simulations de diagrammes de stabilité et $BQ3$ est à 0 V. Plusieurs diagrammes de stabilité ont été simulés pour extraire l'influence des différentes grilles sur les niveaux d'énergie des boîtes. Les résultats sont à la figure 6.2. Le premier diagramme 6.2a est ajusté pour être dans le régime à un électron. Il sert de référence puisque le décalage des lignes de transition est calculé par rapport à ce diagramme. Les points dans les diagrammes montrent quelles valeurs de tension sont utilisées pour le calcul du décalage. Les décalages sont toujours calculés avec deux points de couleur différente, c'est-à-dire entre un point bleu et un point orange.



(**a**) Schéma du dispositif simulé (vue de haut)

(**b**) Schéma du dispositif simulé (vue de côté selon la ligne pointillé de la figure de gauche)

FIGURE 6.1 Dispositif simulé pour tester les grilles virtuelles

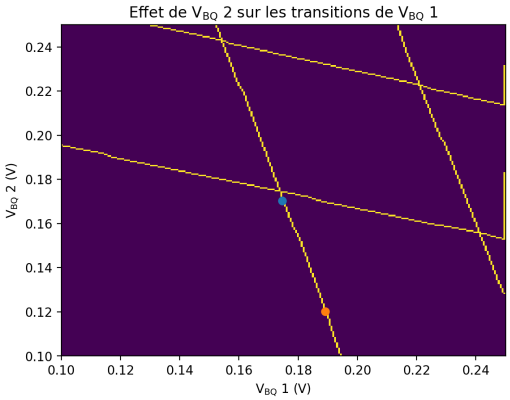
$$\begin{pmatrix} \tilde{V}_{B1} \\ \tilde{V}_{BQ1} \\ \tilde{V}_{B2} \\ \tilde{V}_{BQ2} \\ \tilde{V}_{B3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} & \alpha_{25} \\ 0 & 0 & 1 & 0 & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} & \alpha_{45} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_{B1} \\ V_{BQ1} \\ V_{B2} \\ V_{BQ2} \\ V_{B3} \end{pmatrix} \tag{6.2}$$

$$\tilde{V}_i = \sum_j (CC)_{ij} V_j \tag{6.3}$$

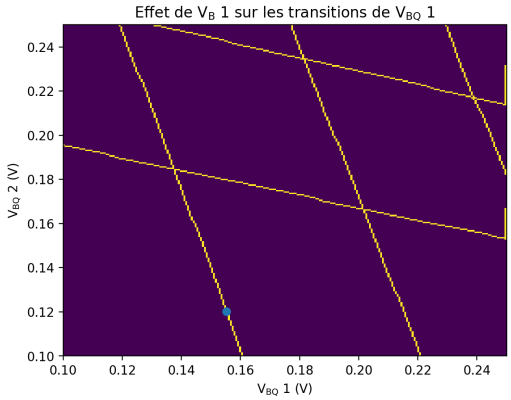
$$V_j = \sum_i (CC^{-1})_{ji} \tilde{V}_i \tag{6.4}$$

Pour montrer comment calculer la matrice de capacités croisées, le calcul de l'élément α_{21} sera détaillé. Cet élément, tel qu'on peut le déduire en regardant le produit matriciel de l'équation 6.2, donne l'influence de la grille $B1$ sur la grille de la boîte 1 $(BQ1)$. Ainsi, les deux diagrammes à utiliser pour calculer α_{21} sont les diagrammes 6.2a et 6.2b. Pour le calcul, la position de la transition de $BQ1$ est mesurée à 0.12 V sur l'axe vertical, mais est choisie de façon arbitraire. Sur le diagramme de référence, la transition se trouve à 0.1891 V sur l'axe horizontal. Après avoir diminué la tension de $B1$ de -0.5 V à -0.4 V, cette même transition se trouve à 0.1552 V sur l'axe horizontal. Ainsi, nous avons le résultat suivant pour α_{21}.

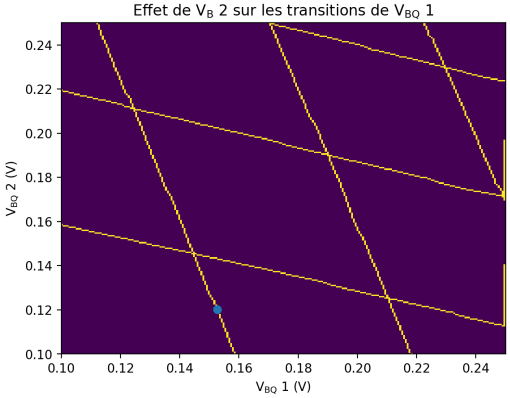
$$\alpha_{21} = \frac{0.1552V - 0.1891V}{-0.4V - (-0.5V)} = \frac{-0.0339V}{0.1V} = -0.339$$



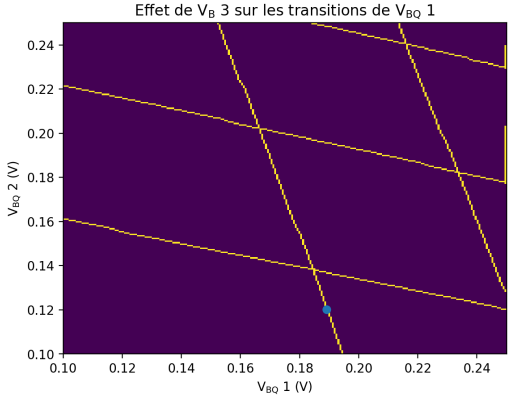
(**a**) Diagramme de référence, toutes les grilles de barrière à -0.5 V et BQ3 à 0 V



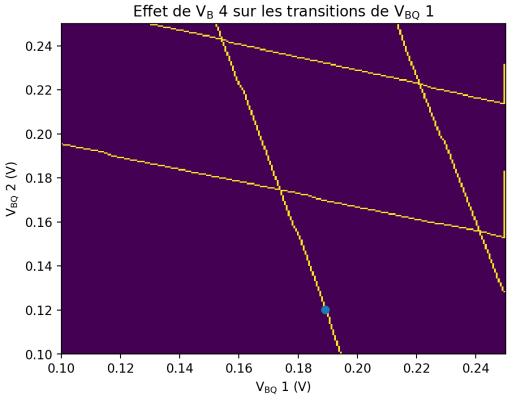
(**b**) B1 à -0.4 V



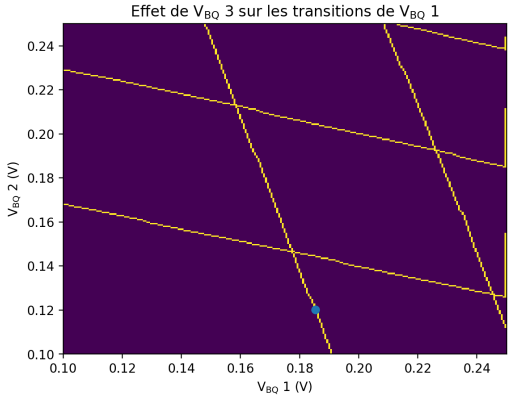
(**c**) B2 à -0.4 V



(**d**) B3 à -0.4 V



(**e**) B4 à -0.4 V



(**f**) BQ3 à 0.1 V

FIGURE 6.2 Diagrammes de stabilité utilisés pour calculer la matrice de capacités croisées. Chaque diagramme diffère légèrement du diagramme de référence lorsqu'une tension de grille est légèrement modifiée par rapport aux tensions initiales ($B1$, $B2$, $B3$, $B4$ à -0.5 V et $BQ3$ à 0 V).

L'opération est ensuite répétée pour chaque élément de la matrice. Cependant, pour mesurer le couplage entre deux grilles qui sont utilisées pour tracer un diagramme de stabilité, il faut utiliser les transitions sur un seul et même diagramme. C'est le cas de l'élément α_{24} qui donne l'effet de $BQ2$ sur $BQ1$. La position de la transition de référence est toujours mesurée à 0.12 V sur l'axe vertical, sauf qu'en variant $BQ2$, on ne fait que suivre la même ligne de transition. Ainsi, pour une variation de $BQ2$ de 0.12 V à 0.17 V, la transition s'est déplacée de 0.1891 V à 0.1747 V. Le calcul de α_{24} donne donc le résultat suivant.

$$\alpha_{24} = \frac{0.1747V - 0.1891V}{0.17V - 0.12V} = \frac{-0.0144V}{0.05V} = -0.288$$

Après avoir calculé tous les éléments de la matrice de capacités croisées qui sont non nuls, l'équation 6.5 est obtenue. L'inverse de cette équation, l'équation 6.6, permet d'utiliser les grilles virtuelles pour faire les diagrammes de stabilité et dicte les tensions à appliquer sur chaque grille pour annuler les couplages entre les grilles. Le même diagramme de stabilité qu'à la figure 6.2a est alors tracé à la figure 6.3 avec les grilles virtuelles. Les transitions sont presque parfaitement droites. Le couplage entre les grilles a donc bien été éliminé. Cependant, la compensation n'est pas parfaite puisqu'une certaine courbure est toujours visible. Bien sûr, une compensation d'ordre plus élevé que le premier ordre pourrait améliorer la compensation, mais compliquerait les calculs pour un gain minime. Par contre, il est à noter que pour avoir une compensation efficace, il faudrait calculer les éléments de matrice en utilisant un δV assez grand pour arriver à bien estimer la pente des transitions. Si une valeur trop petite est utilisée par rapport à la résolution du diagramme, les résultats seront moins bons.

$$\begin{pmatrix} \tilde{V}_{B1} \\ \tilde{V}_{BQ1} \\ \tilde{V}_{B2} \\ \tilde{V}_{BQ2} \\ \tilde{V}_{B3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -0.339 & 1 & -0.364 & -0.288 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -0.275 & -0.370 & 1 & -0.339 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_{B1} \\ V_{BQ1} \\ V_{B2} \\ V_{BQ2} \\ V_{B3} \end{pmatrix} \tag{6.5}$$

$$\begin{pmatrix} V_{B1} \\ V_{BQ1} \\ V_{B2} \\ V_{BQ2} \\ V_{B3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.368 & 1.086 & 0.511 & 0.312 & 0.106 \\ 0 & 0 & 1 & 0 & 0 \\ 0.101 & 0.299 & 0.511 & 1.086 & 0.368 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{V}_{B1} \\ \tilde{V}_{BQ1} \\ \tilde{V}_{B2} \\ \tilde{V}_{BQ2} \\ \tilde{V}_{B3} \end{pmatrix} \tag{6.6}$$

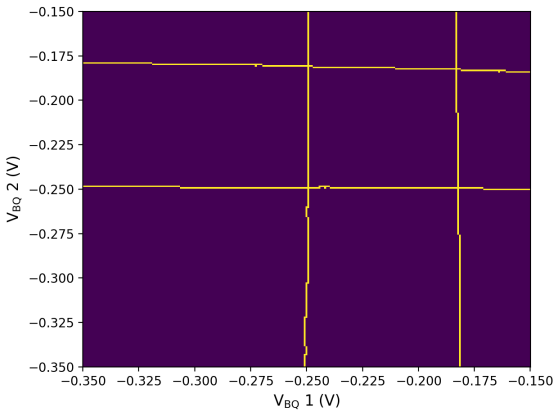


FIGURE 6.3 Diagramme de stabilité de la figure 6.2 a) simulé en utilisant les grilles virtuelles. Les lignes de transitions sont devenues droites, indiquant qu'il n'y a plus de couplage entre les grilles.

6.2 Grilles virtuelles sur FPGA

Une fois que le calcul des grilles virtuelles fonctionne bien dans les simulations, il est ensuite possible de les implémenter avec un FPGA. Il est important de tester un programme avant de l'implémenter avec un FPGA étant donné qu'il est long et parfois difficile de modifier un programme FPGA par la suite. Le travail du programme FPGA ici est seulement de faire le produit matriciel entre la matrice de capacités croisées inverse et les entrées du générateur de signaux arbitraires pour appliquer les bonnes tensions sur les quatre canaux du générateur. Il est donc nécessaire de sauvegarder la matrice dans la mémoire du FPGA et ensuite de programmer le produit matriciel. La sauvegarde de la matrice se fait à l'aide d'un port MEM comme à la figure 6.4. Ce port est une interface mémoire qui permet d'échanger des données entre le FPGA et l'ordinateur. Ainsi, une fois que la matrice est calculée et sauvegardée sur l'ordinateur, il faut la transformer en un vecteur 1D pour la transmettre au FPGA. Le port MEM envoie la matrice dans des registres du FPGA pour qu'elle reste en mémoire. Autrement, le transfert se serait quand même effectué, mais la matrice ne serait restée que quelques cycles d'horloge dans le FPGA. Le bloc FPGA utilisé ici est le bloc «Reg xN» (registre fois N) pour pouvoir sauvegarder tous les éléments de matrice d'un coup. Cependant, les registres de ce bloc sont de 32 bits. Il faut donc ajouter des blocs «Slice» (tronquer) pour convertir les éléments de matrice de 32 bits à 16 bits. On ne garde que les 16 bits les moins significatifs avec cette opération puisque les autres bits ne sont que des zéros qui ne contiennent aucune information. Le bloc «Reg» (registre) peut aussi être utilisé et a l'avantage que la taille en bits du registre peut être spécifiée. Par contre, s'il faut sauvegarder 16 éléments d'une matrice 4x4, il faut 16 blocs «Reg» et faire les connexions nécessaires pour

chacun. Toutes ces connexions alourdiraient énormément le code.

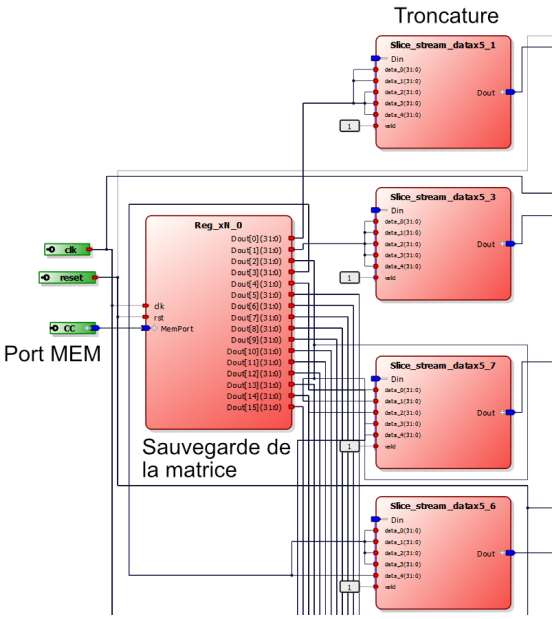


FIGURE 6.4 Sauvegarde de la matrice de capacités croisées dans le FPGA. Le bloc vert CC sert de connexion entre l'ordinateur et le FPGA à travers un port MEM pour transférer la matrice de capacités croisées. Chaque élément de matrice est sauvegardé dans un registre de 32 bits. Puisque les tensions dans l'instrument ont une résolution de 16 bits seulement, les registres sont tronqués pour n'être que 16 bits de long.

La multiplication de matrice se fait en multipliant les registres contenant la matrice de capacités croisées aux signaux d'entrée du générateur de signaux (blocs verts de gauche à la figure 6.5). Les signaux des blocs verts sont convertis au type «dataStreamFCx5» pour réduire le nombre de connexions à faire à chaque étape de calcul. Ce type est compatible avec le type par défaut des signaux qui est «dataStreamMFCx5». La multiplication est ensuite effectuée avec un bloc «Multiplier» (multiplicateur) entre chaque élément de la matrice et le signal d'entrée correspondant. Il faut également tronquer le résultat de la multiplication puisque le résultat dépasse 16 bits. Pour chaque rangée de la matrice, on additionne le résultat des multiplications avec un bloc «Adder» (additionneur) pour donner le résultat final. La figure 6.5 montre le calcul pour la première rangée seulement pour alléger l'image, mais cette portion est en réalité répétée quatre fois, une fois pour chaque rangée de la matrice. L'équation 6.7 montre les éléments impliqués dans la multiplication de la figure 6.5.

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix} = \begin{pmatrix} \alpha_{11} V_1 + \alpha_{12} V_2 + \alpha_{13} V_3 + \alpha_{14} V_4 \\ \alpha_{21} V_1 + \alpha_{22} V_2 + \alpha_{23} V_3 + \alpha_{24} V_4 \\ \alpha_{31} V_1 + \alpha_{32} V_2 + \alpha_{33} V_3 + \alpha_{34} V_4 \\ \alpha_{41} V_1 + \alpha_{42} V_2 + \alpha_{43} V_3 + \alpha_{44} V_4 \end{pmatrix} \tag{6.7}$$

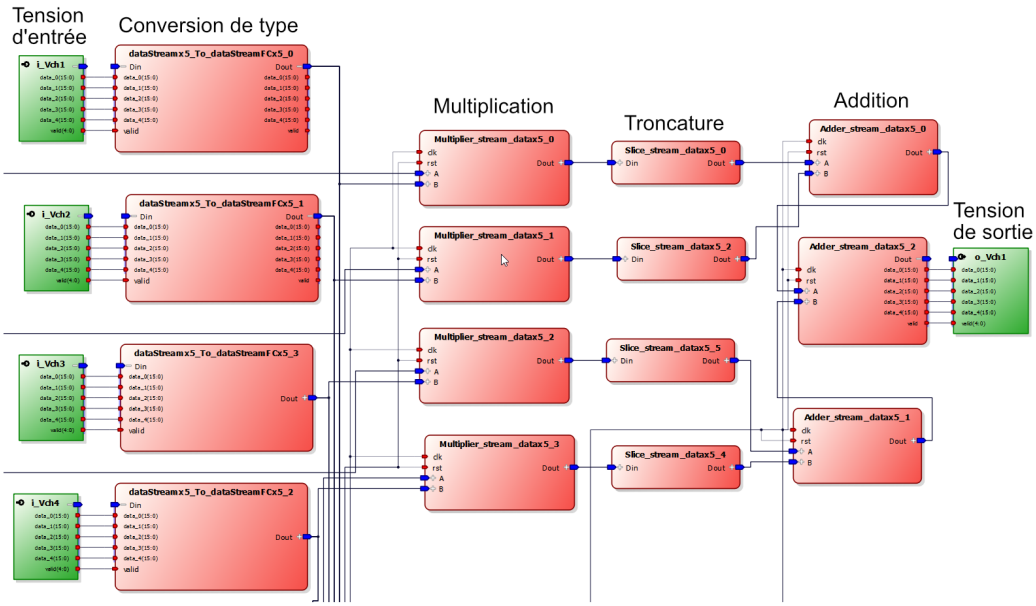


FIGURE 6.5 Multiplication matricielle avec FPGA flow. Les registres de la figure 6.4 sont connectés aux blocs multiplicateurs, mais ne sont pas visibles dans la figure. Les signaux de tension arrivent à l'entrée du programme dans les blocs verts de gauche et sont convertis au type «dataStreamFCx5» pour réduire le nombre de connexion à faire dans le programme. Ensuite, les éléments de matrice de la figure 6.4 sont multipliés aux signaux de tension. Les résultats sont tronqués puis additionnés pour donner le signal des grilles virtuelles pour le premier canal du générateur de signaux. Cette partie de programme FPGA flow effectue le produit matricielle de l'équation 6.7.

Le programme précédent peut gérer jusqu'à quatre grilles en même temps, car il y a quatre canaux sur un générateur de signaux arbitraires. La figure 6.6 montre la différence entre le calcul fait par ordinateur et par FPGA. Il y a une légère différence entre les deux résultats, mais cela vient d'un décalage de quelques millivolts à la sortie du générateur de signaux par rapport au zéro de tension. Pour augmenter le nombre de grilles virtuelles, il faut pouvoir transmettre le signal des grilles qui sont balayées aux autres générateurs de signaux et synchroniser les calculs de tension à appliquer. Cependant, la connectivité entre les différents appareils est surtout faite pour transmettre des signaux de synchronisation et donc peu de données. La transmission de signaux de 16 bits a alors besoin de se faire en

plusieurs transferts, ce qui complique la communication et la ralentit. Une alternative plutôt simple est de tout simplement prendre le code de rampe qui calcule la tension à appliquer, de le copier sur chaque générateur ayant besoin des grilles virtuelles puis de l'exécuter sur ces derniers. HVI peut ensuite s'assurer que tous les générateurs sont synchronisés et envoyer les valeurs de tension dans FPGA flow à l'aide d'un port MEM. Cette solution devrait fonctionner, mais n'a pas encore été complètement testée. Pour utiliser plus de grilles virtuelles, il faut aussi pouvoir faire un plus gros produit matriciel. Naïvement, la solution serait de calculer le produit matriciel sur le FPGA de chaque générateur utilisé. Par contre, déjà avec deux générateurs (8 canaux), il faut gérer 64 éléments de matrice comparativement à seulement 16 pour quatre canaux. Le programme FPGA devient donc rapidement beaucoup plus gros. Heureusement, comme on peut le voir à l'équation 6.7, la première tension de grille ne dépend que des éléments de matrice qui sont sur la première rangée. Ainsi, pour un générateur de quatre canaux, il ne faut réellement utiliser que $4 \times 8 = 32$ éléments de matrice pour calculer le produit de matrice associé à 8 grilles virtuelles. Ainsi, la taille du programme croît linéairement avec le nombre de grilles virtuelles au lieu de grossir de façon quadratique. Beaucoup de temps est donc économisé, en ce qui concerne la programmation, et les ressources utilisées par le FPGA sont aussi réduites en ne calculant pas le produit de matrice au complet. Finalement, tant qu'il y a assez de mémoire dans le FPGA de tous les générateurs de signaux utilisés pour sauvegarder la matrice de capacités croisées et qu'une synchronisation est possible entre ces derniers, le nombre de grilles virtuelles pouvant être utilisées peut être augmenté pour accommoder les dispositifs avec un nombre grandissant de qubits. Par contre, le programme qui doit gérer les grilles virtuelles devient de plus en plus lourd à mesure que des grilles sont ajoutées, ce qui demande de faire plus de programmation.

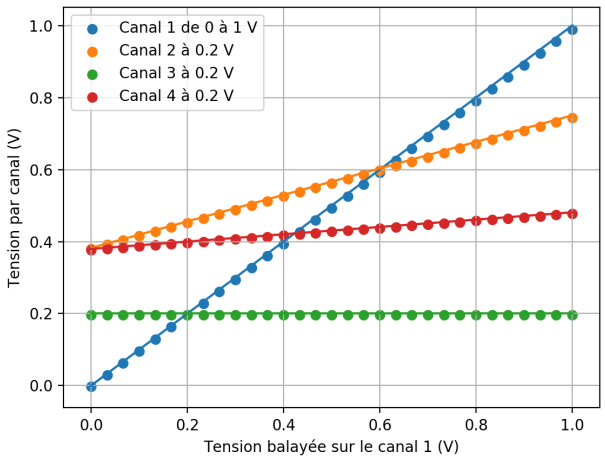


FIGURE 6.6 Comparaison du signal de sortie entre des grilles virtuelles simulées et calculées avec le FPGA. Les points sont les données expérimentales et les lignes droites sont les simulations. La première grille est balayée de 0 à 1 V, les trois autres grilles sont à 0.2 V. La matrice utilisée pour le calcul est la même qu'à l'équation 6.6 sauf que la 5e rangée et la 5e colonne ont été enlevées pour avoir une matrice 4x4.

Chapitre 7

Perspectives

7.1 Problèmes à résoudre

Malgré les résultats encourageants obtenus durant ce projet, il reste encore quelques problèmes à régler. Premièrement, les grilles virtuelles ne fonctionnent pour l'instant qu'avec des tensions positives. Il s'agit bien sûr d'une grande limitation puisqu'il faut pouvoir utiliser des tensions négatives pour créer le confinement des boîtes quantiques. La source probable du problème a toutefois déjà été identifiée. Le programme de multiplication FPGA utilise la multiplication de nombres non signés pour calculer le produit matriciel. La multiplication signée n'a pas été utilisée dès le départ puisqu'elle créait des erreurs dans la compilation du programme. Le problème est cependant déjà en voie d'être résolu en utilisant des blocs de multiplication utilisant des nombres signés. Ce changement nécessitera l'ajustement des connexions entre les blocs de multiplication et de troncature, puisque le calcul des multiplications se fait différemment. Comme il a été mentionné dans la section précédente, il reste aussi à tester la communication entre différents générateurs de signaux pour pouvoir augmenter le nombre de grilles virtuelles à 8 ou plus.

Deuxièmement, pour faire un diagramme de stabilité complet, il faut pouvoir sauvegarder idéalement autour d'un million de points pour un diagramme de 1000 x 1000 points par exemple. Le digitaliseur ne permet pour le moment que de sauvegarder un peu plus de 1000 points en mémoire, mais cette limite sera devrait être augmentée à un million de points grâce à une mise à jour de microprogramme («firmware» en anglais) par Keysight. Cela évitera de devoir séparer un diagramme en plusieurs diagrammes plus petits.

Troisièmement, la plus récente version du programme qui utilise seulement HVI, n'est

pas encore entièrement fonctionnelle. Il est possible de faire des rampes 1D, mais pas des diagrammes en 2D. La boucle de rampes contenues dans le diagramme 2D n'est pas tout à fait bien exécutée et trop de rampes sont lancées pour un diagramme donné. Ce problème est probablement dû à des erreurs dans la programmation HVI et une correction est présentement recherchée.

Finalement, il n'y a toujours pas d'interface conviviale pour utiliser tous les programmes décrits dans ce mémoire. Il est compliqué de faire une seule interface lorsque les programmes sont en développement constant et que plusieurs fonctionnalités s'ajoutent avec le temps. Une interface Labber a été faite pour la version VHDL faite par Larissa Njejimana du programme qui mesure les diagrammes de stabilité. Par contre, cette version n'est pas très intuitive et contient plusieurs bogues. Pour ces raisons et à cause des nombreuses modifications apportées au programme, cette interface n'a pas été mise à jour.

7.2 Suite du projet

À plus long terme, il est prévu d'ajouter plusieurs fonctionnalités au «Quantum Engineering Toolkit» (QET). Une fois les routines de caractérisation complétées, il sera possible d'implémenter les routines de contrôle de qubits. Il sera donc possible de faire des portes logiques et d'évaluer leur fidélité avec du «randomized benchmarking». Comme il a été discuté dans la section 2.5, plusieurs algorithmes ont été développés pour l'initialisation automatique des boîtes quantiques. Un projet en collaboration avec le professeur Félix Camirand Lemyre au département de mathématiques de l'Université de Sherbrooke vise à améliorer l'algorithme développé par Maxime Lapointe-Major à l'aide de techniques d'apprentissage machine [38] [39]. Le travail de Marc-Antoine Genest a déjà été une étape importante pour atteindre cet objectif. Une fois tous ces éléments en place, le QET sera une plateforme versatile et efficace pour la caractérisation, mais aussi la manipulation cohérente des qubits de spin. Pour tirer profit de la performance de la plateforme, il est prévu de tester des échantillons de boîtes quantiques fabriqués entièrement avec des procédés de fabrication industriels. Cette méthode permet d'avoir un approvisionnement continu d'échantillons qui ont des propriétés reproductibles aux températures cryogéniques [40]. Avec l'utilisation d'un interposeur développé spécifiquement pour les processeurs quantiques [41], le rendement des expériences devrait être considérablement amélioré. Ainsi, la recherche et le développement d'un processeur quantique basé les qubits de spin pourront alors être grandement accélérés.

Conclusion

Ce projet a permis d'explorer les capacités du «Quantum Engineering Toolkit» de Keysight Technologies en ce qui concerne la caractérisation et la manipulation des qubits de spin. Bien que cette plateforme ne réponde pas présentement à tous les besoins techniques imposés par les qubits de spin, surtout pour la manipulation cohérente, une seconde version du QET est en préparation chez Keysight pour mieux répondre aux besoins spécifiques aux qubits de spin. Malgré cela, la plateforme est déjà un outil attrayant pour caractériser des boîtes quantiques. Le grand nombre de canaux dans un format compact rend cette solution modulaire pour permettre l'intégration future de plus en plus de qubits. De plus, les FPGAs de la plateforme la rendent performante et flexible pour faire les différentes routines de caractérisations typiques aux qubits de spin.

Comme preuve de concept des capacités du QET, la mesure d'un diagramme de stabilité a été effectuée sur un transistor pour comparer sa précision à celle d'un SMU. Pour un temps d'intégration de $80\,\mu s$, la précision des deux appareils est similaire. Par contre, pour une plage de tension identique, une mesure faite avec le FPGA est plus de 1000 fois plus rapide qu'une mesure faite avec un SMU. Pour tirer profit de la vitesse de mesure du FPGA, une acquisition en mode vidéo a été faite. Pour un diagramme avec une plage de -15 mV à 15 mV (31 points) pour l'axe des x et de 1 à 1.15 V (31 points) pour l'axe des y, le temps d'acquisition est de 3.5 millisecondes sans enregistrement. Si les données sont enregistrées avec Numpy, le temps de mesure augmente à 30 ms, ce qui donne un taux de rafraîchissement de 33.3 IPS.

De plus, le concept de grille virtuelle a été implémenté dans la plateforme pour permettre d'enlever le couplage capacitif entre les grilles voisines d'un échantillon de boîte quantique. Pour l'instant, il n'est possible d'utiliser que quatre grilles virtuelles et que des tensions positives. Par contre, il est prévu d'augmenter le nombre de grilles virtuelles à 8 et plus et également de permettre l'utilisation de tensions négatives très prochainement.

Finalement, une nouvelle version d'un programme pour mesurer des diagrammes de stabilité a été développée. Cette nouvelle version, codée entièrement dans HVI, facilitera l'ajout de nouvelles fonctionnalités telles que les grilles virtuelles pour la mesure de diagrammes de stabilité. Il reste quelques bogues à régler pour que cette nouvelle version puisse être utilisée, mais cela devrait aussi être terminé très prochainement.

Une fois que toutes les fonctions de caractérisation de base seront prêtes à être utilisées avec le QET, des fonctions plus complexes seront ajoutées. Par exemple, un algorithme basé sur l'apprentissage machine pourra initialiser automatiquement une boîte quantique dans le régime à un électron. De plus, des routines de contrôle de qubit pourront être incluses à la plateforme pour pouvoir faire des expériences plus complètes sur des qubits de spin.

Annexe A

Codes HVI

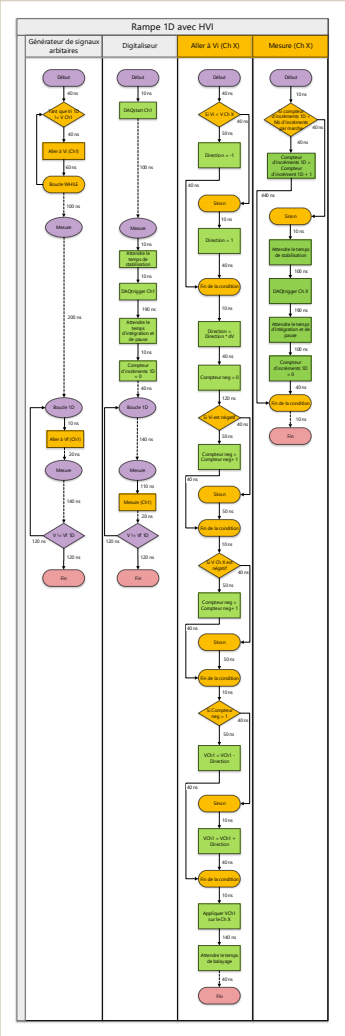


FIGURE A.1 Schéma du programme HVI pour faire une rampe 1D. Cette figure est vectorielle est devrait être lue en format électronique pour être lisible.

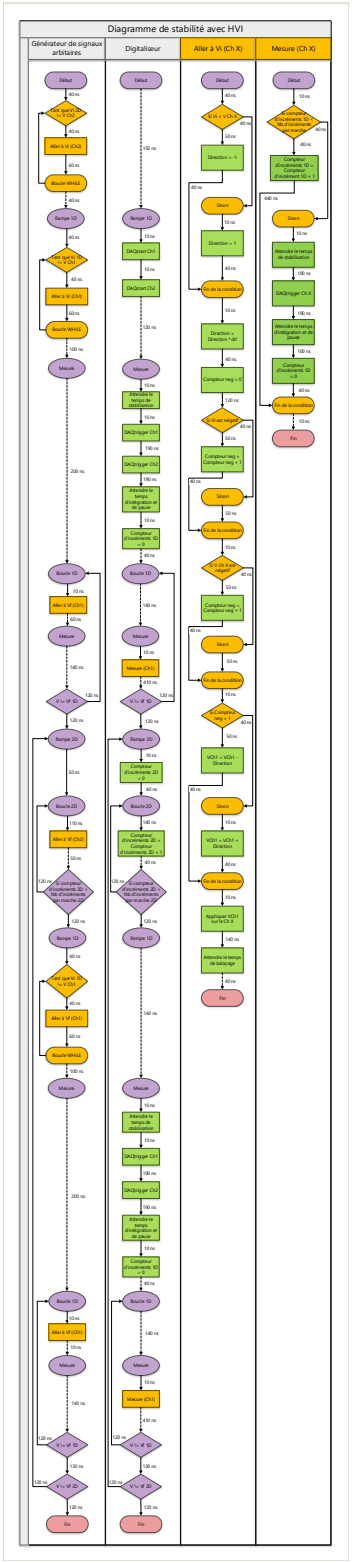


FIGURE A.2 Schéma du programme HVI pour faire un diagramme de stabilité. Cette figure est vectorielle est devrait être lue en format électronique pour être lisible.

N°	Nom du registre	Usage
1	Vi 1D	Param.
2	Vf 1D	Param.
3	Temps de balayage 1D	Param.
4	Vi 2D	Param.
5	Vf 2D	Param.
6	Temps de balayage 2D	Param.
7	Registre 6	N/A
8	Registre 7	N/A
9	Registre 8	N/A
10	Direction	Variable
11	Compteur neg	Variable
12	Firmware	Info
13	Tension Ch1	Info
14	Tension Ch2	Info
15	Tension Ch3	Info
16	Tension Ch4	Info

(**a**) Registres du générateur de signaux arbitraires

N°	Nom du registre	Usage
1	Compteur d'incréments 1D	Variable
2	Nombres d'incréments par marche 1D	Param.
3	Nombres d'incréments par marche 2D	Param.
4	Temps de stabilisation	Param.
5	Temps d'intégration	Param.
6	Compteur d'incréments 2D	Variable
7	Registre 6	N/A
8	Registre 7	N/A
9	Registre 8	N/A
10	Registre 9	N/A
11	Registre 10	N/A
12	Firmware	Info
13	Registre 12	N/A
14	Registre 13	N/A
15	Registre 14	N/A
16	Registre 15	N/A

(**b**) Registres du digitaliseur

Table A.1 Attribution des paramètres nécessaires à l'exécution du programme de la figure A.2 pour chaque registre du générateur de signaux arbitraires et du digitaliseur. Les paramètres doivent être initialisées avant l'exécution du programme. Les variables, quant à elles, servent de mémoire pour le programme et n'ont pas besoin d'être initialisées. Les registres qui n'ont pas de nom personnalisé, eux, ne sont pas utilisés. Les registres «info» fournissent des informations pertinentes lors de l'exécution du programme. Le registre «Firmware» enregistre un nombre associé à la version du programme FPGA flow présentement installé dans chaque instrument. Ce nombre arbitraire est défini et modifié par la personne utilisant le programme pour pouvoir savoir en tout temps quelle version est utilisée présentement. De plus, les registres de tension sont mis à jour lorsque la tension d'un canal est changée pour que la tension de l'instrument puisse être lue au besoin.

Bibliographie

[1] P.W. Shor. Dans *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134. IEEE Comput. Soc. Press, (1994).

[2] Carl Pomerance. *NOTICES AMER. MATH. SOC* **43**, 1473—-1485 (1996).

[3] Kelly et Julian. Dans *Bulletin of the American Physical Society*, volume 63 (American Physical Society, Los Angeles, 2018).

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G.S.L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, et John M. Martinis. *Nature* **574**, 505–510 (2019).

[5] D. P. Franke, J. S. Clarke, L. M.K. Vandersypen, et M. Veldhorst. *Microprocessors and Microsystems* **67**, 1–7 (2019).

[6] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, et R. J. Schoelkopf. *Physical Review A - Atomic, Molecular, and Optical Physics* **76**(4) (2007).

[7] J. I. Cirac et P. Zoller. *Physical Review Letters* **74**(20), 4091–4094 (1995).

[8] Jason Alicea, Yuval Oreg, Gil Refael, Felix Von Oppen, et Matthew P.A. Fisher. *Nature Physics* **7**(5), 412–417 (2011).

[9] Daniel Loss et David P. DiVincenzo. *Physical Review A* **57**(1), 120–126 (1998).

[10] B. E. Kane. *Nature* **393**(6681), 133–137 (1998).

[11] C. H. Yang, R. C. C. Leon, J. C. C. Hwang, A. Saraiva, T. Tanttu, W. Huang, J. Camirand Lemyre, K. W. Chan, K. Y. Tan, F. E. Hudson, K. M. Itoh, A. Morello, M. Pioro-Ladrière, A. Laucht, et A. S. Dzurak. (2019).

[12] D. M. Zajac, A. J. Sigillito, M. Russ, F. Borjans, J. M. Taylor, G. Burkard, et J. R. Petta. *Science* **359**(6374) (2018).

[13] S. Rochette, M. Rudolph, A. M. Roy, M. J. Curry, G. A.Ten Eyck, R. P. Manginell, J. R. Wendt, T. Pluym, S. M. Carr, D. R. Ward, M. P. Lilly, M. S. Carroll, et M. Pioro-Ladrière. *Applied Physics Letters* **114**(8) (2019).

[14] Bent Weber, Suddhasatta Mahapatra, Thomas F. Watson, et Michelle Y. Simmons. *Nano Letters* **12**(8), 4001–4006 (2012).

[15] W. G. Van der Wiel, S. De Franceschi, J. M. Elzerman, T. Fujisawa, S. Tarucha, et L. P. Kouwenhoven. (2003).

[16] Tim Botzem, Michael D. Shulman, Sandra Foletti, Shannon P. Harvey, Oliver E. Dial, Patrick Bethke, Pascal Cerfontaine, Robert P.G. McNeil, Diana Mahalu, Vladimir Umansky, Arne Ludwig, Andreas Wieck, Dieter Schuh, Dominique Bougeard, Amir Yacoby, et Hendrik Bluhm. *Physical Review Applied* **10**(5) (2018).

[17] F. H. L. Koppens, C. Buizert, K. J. Tielrooij, I. T. Vink, K. C. Nowack, T. Meunier, L. P. Kouwenhoven, et L. M. K. Vandersypen. *Nature* **442**(7104), 766–771 (2006).

[18] Ryan M. Jock, N. Tobias Jacobson, Patrick Harvey-Collard, Andrew M. Mounce, Vanita Srinivasa, Dan R. Ward, John Anderson, Ron Manginell, Joel R. Wendt, Martin Rudolph, Tammy Pluym, John King Gamble, Andrew D. Baczewski, Wayne M. Witzel, et Malcolm S. Carroll. *Nature Communications* **9**(1), 1–8 (2018).

[19] J. Medford, J. Beil, J. M. Taylor, E. I. Rashba, H. Lu, A. C. Gossard, et C. M. Marcus. *Physical Review Letters* **111**(5), 050501 (2013).

[20] J. P. Dehollain, J. J. Pla, E. Siew, K. Y. Tan, A. S. Dzurak, et A. Morello. *Nanotechnology* **24**(1) (2013).

[21] M. Pioro-Ladrière, T. Obata, Y. Tokura, Y. S. Shin, T. Kubo, K. Yoshida, T. Taniyama, et S. Tarucha. *Nature Physics* **4**(10), 776–779 (2008).

[22] R. Hanson, L. H.Willems Van Beveren, I. T. Vink, J. M. Elzerman, W. J.M. Naber, F. H.L. Koppens, L. P. Kouwenhoven, et L. M.K. Vandersypen. *Physical Review Letters* **94**(19), 1–4 (2005).

[23] Andrea Morello, Jarryd J. Pla, Floris A. Zwanenburg, Kok W. Chan, Kuan Y. Tan, Hans Huebl, Mikko Möttönen, Christopher D. Nugroho, Changyi Yang, Jessica A. Van Donkelaar, Andrew D.C. Alves, David N. Jamieson, Christopher C. Escott, Lloyd C.L. Hollenberg, Robert G. Clark, et Andrew S. Dzurak. *Nature* **467**(7316), 687–691 (2010).

[24] Andrew Mounce, Phillip Lewis, Cara Monical, N. Tobias Jacobson, Albert Grine, Martin Rudolph, John Anderson, Joel Wendt, Tammy Pluym, Dan Ward, Kurt Larson, Michael Lilly, et Malcolm Carroll. Dans *Bulletin of the American Physical Society*, volume 64. American Physical Society, (2019).

[25] Maxime Lapointe-Major. Mémoire de Maîtrise, Université de Sherbrooke, (2018).

[26] Marc-Antoine Genest. Mémoire de Maîtrise, Université de Sherbrooke, (2019).

[27] T. A. Baart, P. T. Eendebak, C. Reichl, W. Wegscheider, et L. M. K. Vandersypen. *Applied Physics Letters* **108**(21), 213104 (2016).

[28] D. T. Lennon, H. Moon, L. C. Camenzind, Liuqi Yu, D. M. Zumbühl, G. A.D. Briggs, M. A. Osborne, E. A. Laird, et N. Ares. *npj Quantum Information* **5**(1) (2019).

[29] J. Stehlik, Y. Y. Liu, C. M. Quintana, C. Eichler, T. R. Hartke, et J. R. Petta. *Physical Review Applied* **4**(1) (2015).

[30] Gang Huang, Yilun Xu, Lawrence Doolittle, Unpil Baek, et Irfan Siddiqi. Dans *Bulletin of the American Physical Society*, volume 64. American Physical Society, (2019).

[31] Unpil Baek, Yilun Xu, Gang Huang, Lawrence Doolittle, et Irfan Siddiqi. Dans *Bulletin of the American Physical Society*, volume 64. American Physical Society, (2019).

[32] J. M. Elzerman, R. Hanson, J. S. Greidanus, L. H. Willems van Beveren, S. De Franceschi, L. M.K. Vandersypen, S. Tarucha, et L. P. Kouwenhoven. *Physical Review B* **67**(16) (2003).

[33] Julien Camirand Lemyre, Michel Pioro-Ladrière, et Sophie Rochette. *Specification of control signals for spin qubits*. Rapport technique, Université de Sherbrooke, Sherbrooke, (2019).

[34] Larissa Njejimana. *Lock-In Project*. Rapport technique, Université de Sherbrooke, Sherbrooke, (2018).

[35] Larissa Njejimana. *Sweeper IP Project*. Rapport technique, Université de Sherbrooke, Sherbrooke, (2018).

[36] Paul. Read et Mark-Paul. Meyer. *Restoration of Motion Picture Film*. Butterworth-Heinemann, (2000).

[37] C. Volk, A. M.J. Zwerver, U. Mukhopadhyay, P. T. Eendebak, C. J. van Diepen, J. P. Dehollain, T. Hensgens, T. Fujita, C. Reichl, W. Wegscheider, et L. M.K. Vandersypen. *npj Quantum Information* **5**(1), 1–8 (2019).

[38] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, et Yoshua Bengio. Dans *Advances in Neural Information Processing Systems 27*, Z Ghahramani, M Welling, C Cortes, N D Lawrence, et K Q Weinberger, 2672–2680. Curran Associates, Inc. (2014).

[39] Sandesh S. Kalantre, Justyna P. Zwolak, Stephen Ragole, Xingyao Wu, Neil M. Zimmerman, M. D. Stewart, et Jacob M. Taylor. *npj Quantum Information* **5**(1), 1–10 (2019).

[40] Ph. Galy, J. Camirand Lemyre, P. Lemieux, F. Arnaud, D. Drouin, et Michel Pioro-Ladriere. *IEEE Journal of the Electron Devices Society* **6**(April), 594–600 (2018).

[41] J. H. Béjanin, T. G. McConkey, J. R. Rinehart, C. T. Earnest, C. R.H. McRae, D. Shiri, J. D. Bateman, Y. Rohanizadegan, B. Penava, P. Breul, S. Royak, M. Zapatka, A. G. Fowler, et M. Mariantoni. *Physical Review Applied* **6**(4) (2016).